(2)

NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA

THESIS

DESIGN AND MONTE CARLO ANALYSIS OF AN UNMANNED AERIAL VEHICLE

by

Joseph Paul Fordham

December 1993

Thesis Advisor: Isaac Kaminer

Approved for public release; distribution is unlimited

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1993	Master's Thesis 4/93-10/93

4. TITLE AND SUBTITLE
Design and Monte Carlo Analysis of an Unmanned Aerial Vehicle

5. FUNDING NUMBERS

6. AUTHOR(S)
Joseph Paul Fordham

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, California 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

In the last several years, software innovations and the increasing speed and availability of microcomputers and workstations have made the dynamic simulation of complex systems more practical. One such system, a short-range Unmanned Aerial Vehicle called Bluebird, was previously modeled on Simulink, a commercial software package. The high fidelity model includes six degree of freedom nonlinear equations of motion with onboard sensors and a Global Positioning System and inertial navigation system.

Because of interest expressed by the Unmanned Aerial Vehicle Joint Program Office in how accurately a UAV could identify a target's geographical coordinates, the Bluebird model, with an added guidance and control system, was evaluated as to its navigational and attitudinal accuracy in a dynamic simulation using Monte Carlo techniques. Because of the modular nature of the simulation, future evaluations of manned or unmanned aircraft and avionics will involve relatively uncomplicated changes to the existing model.

14. SUBJECT TERMS	15. NUMBER OF PAGES
	89
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Approved for public release; distribution is unlimited.

DESIGN AND MONTE CARLO ANALYSIS OF AN UNMANNED AERIAL VEHICLE

by

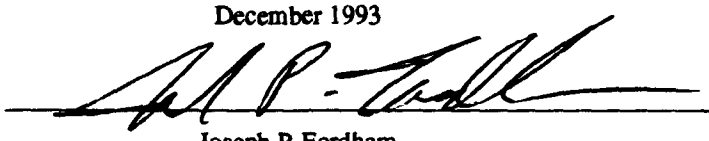
Joseph P. Fordham
Lieutenant, United States Navy
B.S., United States Naval Academy, 1985

Submitted in partial fulfillment
of the requirements for the degree of

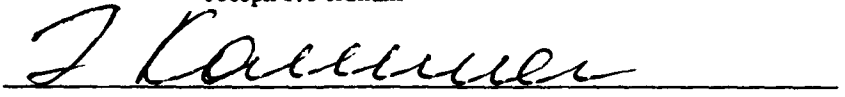
MASTER OF SCIENCE IN AERONAUTICAL ENGINEERING

from the

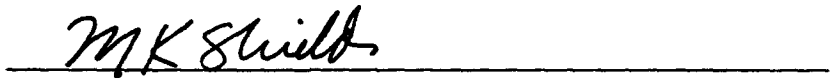
NAVAL POSTGRADUATE SCHOOL
December 1993

Author: 

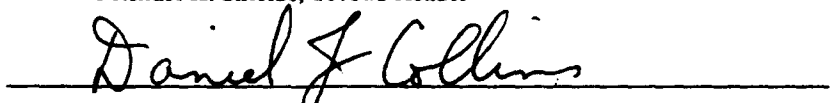
Joseph P. Fordham

Approved by: 

Isaac I. Kaminer, Thesis Advisor



Michael K. Shields, Second Reader



Daniel J. Collins, Chairman
Department of Aeronautics and Astronautics

ABSTRACT

In the last several years, software innovations and the increasing speed and availability of microcomputers and workstations have made the dynamic simulation of complex systems more practical. One such system, a short-range Unmanned Aerial Vehicle called Bluebird, was previously modeled on Simulink, a commercial software package. The high fidelity model includes six degree of freedom nonlinear equations of motion with onboard sensors and a Global Positioning System and inertial navigation system.

Because of interest expressed by the Unmanned Aerial Vehicle Joint Program Office in how accurately a UAV could identify a target's geographical coordinates, the Bluebird model, with an added guidance and control system, was evaluated as to its navigational and attitudinal accuracy in a dynamic simulation using Monte Carlo techniques. Because of the modular nature of the simulation, future evaluations of manned or unmanned aircraft and avionics will involve relatively uncomplicated changes to the existing model.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

In the last several years, the increasing speed and availability of microcomputers and workstations has allowed complex simulation of dynamic systems which previously could only be practically evaluated at static points in time. In order to rigorously evaluate the performance of a nonlinear multiple input, multiple output (MIMO) modeled air vehicle, such as an unmanned aerial vehicle (UAV), it is necessary to dynamically simulate it. This is due to the complex way in which errors are propagated throughout a nonlinear closed loop system. The most common way to dynamically evaluate errors of a system subject to noise is using the Monte Carlo simulation method, which is essentially the repetition of the same experiment over many times.

To support the ongoing Naval Postgraduate School UAV effort, a high fidelity UAV computer model, incorporating aircraft equations of motion and sensors, guidance, navigation, and control models, was constructed utilizing the thesis work of two former Naval Postgraduate School students. The primary goal was an analysis of how well a UAV could designate a fixed ground target considering the reliability of its onboard sensors. To achieve this, the model, with a simulated FLIR (Forward Looking Infrared Receiver) or low light television payload attached, was run over a fixed track on a reconnaissance mission. For the Monte Carlo simulation and analysis, sensor error data was taken from a Naval Air Development Center (NADC) report, "Unmanned Aerial Vehicle Flight Management System Error Analysis" [Ref. 1] and added to the UAV model. The Monte Carlo analysis per-

formed on this model was intended to complement [Ref. 1], which analyzed and compared targeting errors for several distinct UAV architectures, including close-range, short-range, and medium-range vehicles performing a similar mission. This thesis is an extension of the UAV Error Analysis Report in that it considers how sensor and measurement errors propagate through a UAV guidance, navigation, and control system as it directs the vehicle along a given trajectory. This report shows the importance of considering the feedback nature of a controlled system in the error analysis of its performance.

Consider the system shown in Figure 1. Here the aircraft block contains actuators and sensors which are subject to measurement and position errors. Errors from one sensor used by the control system for feedback propagate into all feedback channels in a complex nonlinear manner. For example, a change in roll rate may cause a false measurement in the pitch rate gyro. If this gyro is being used to help determine aircraft attitude, this false measurement will cause an inaccuracy in measured attitude. To restore proper attitude, an actual pitch rate will be created, which may cause a false measurement of roll and yaw rates and correspondingly of roll and yaw angles. It is due to the nonlinear, feedback nature of these phenomena that simple statistical methods applied to a static model of a vehicle cannot be used to accurately evaluate how well a sensor suite measures aircraft flight parameters. In the analysis presented here, consideration was also given to sensor dynamics (how sensitive sensors are to a rapidly changing input). Angular sensors on the aircraft and lookdown and azimuth errors for the FLIR/ Low Light Television sensor were those specified for the MIAG architecture in [Ref. 1]. To complete the analysis, Monte Carlo simulation was implemented with the NPS UAV model by repeatedly

running the model over a fixed track and collecting and analyzing the onboard sensor data. The targeting errors obtained given the advertised sensor accuracies were then compared with the targeting errors determined with Monte Carlo simulation.

To accomplish the error analysis, the UAV model developed by two prior thesis students had to be completed. Figure 1 shows how their work fits into the complete model. The model consisted of aircraft nonlinear equations of motion and sensor models developed by Kuechenmeister [Ref. 2] and a high fidelity Differential Global Positioning System and inertial navigation system model created by Marquis [Ref. 3]. A linear quadratic regulator, whose design and implementation is discussed in Chapter III, was constructed to stabilize the model and provide a means to control it in heading, airspeed, and attitude in steady state. To steer the model along a fixed course, a waypoint guidance feature was also developed. While the entire model is not representative of a particular UAV, the way that its guidance, navigation, and control are interrelated in their operation is indicative of how a UAV equipped with an integrated sensor package, like the new MIAG (modular integrated avionics group) concept, can be sensitive to sensor errors.

The next chapter outlines the components of the computer model. Chapter III covers in greater depth the coontributions to the model made by the author, including controller and waypoint guidance design and payload modeling. Chapter IV explains the Monte Carlo analysis used in analyzing the model's targeting errors. Chapter V discusses how the simulation was performed and Chapters VI and VII include results of the analysis and conclusions.

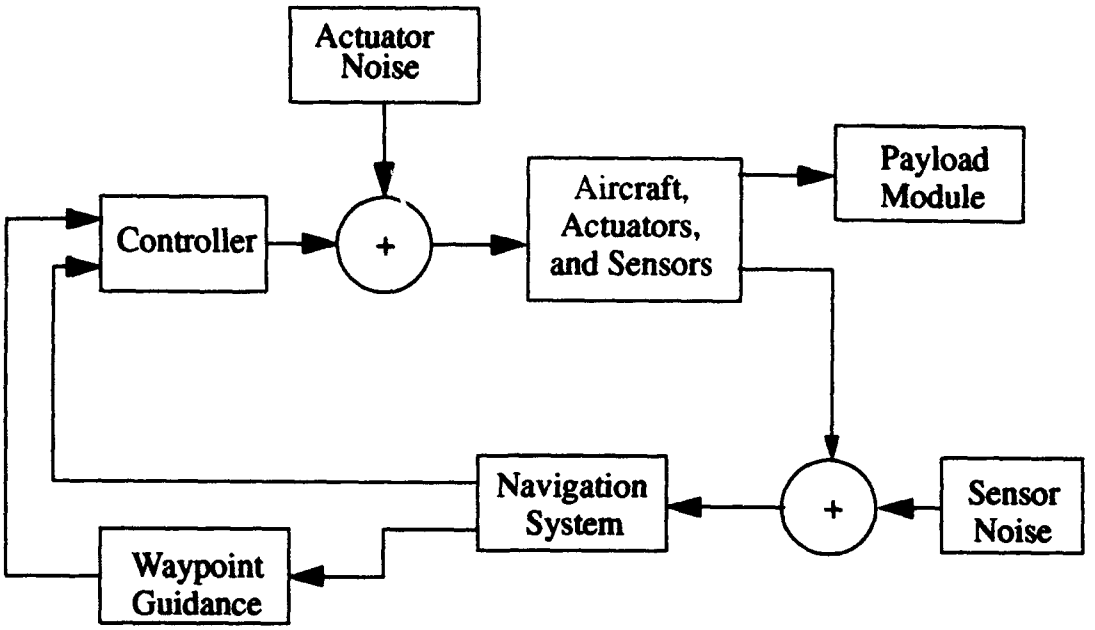


Figure 1. System Overview

II. THE UAV MODEL

The complete model used in this thesis is shown in Figure 2. It is composed of several distinct subsystems, some of which were constructed as parts of previous theses by Naval Postgraduate School Students. Kuechenmeister was responsible in a large part for the aircraft equations of motion and the sensor models [Ref. 2]. Marquis constructed the six satellite differential Global Positioning System (GPS) model used in the simulation as well as a Kalman Filter integrating the GPS model with an inertial navigation system [Ref. 3]. The model, which was constructed using Simulink, a program commercially available from Mathworks, Inc., is designed to run on either a Sun workstation or a personal computer running Matlab for Windows with Simulink. Because of the complexity of the simulation, it should be run on the fastest platform available. As a benchmark, the entire simulation running on a Sun Sparc10 workstation with 64 megabytes of RAM runs at a rate of about one second of real time for every five minutes of simulation time.

A. EQUATIONS OF MOTION AND SENSOR MODELS

The UAV used in the simulation is called Bluebird. The actual aircraft is a high-wing monoplane with a wingspan of 12.4 feet and a weight of 55 pounds. The aircraft is controlled using servos which actuate conventional elevator, aileron, and

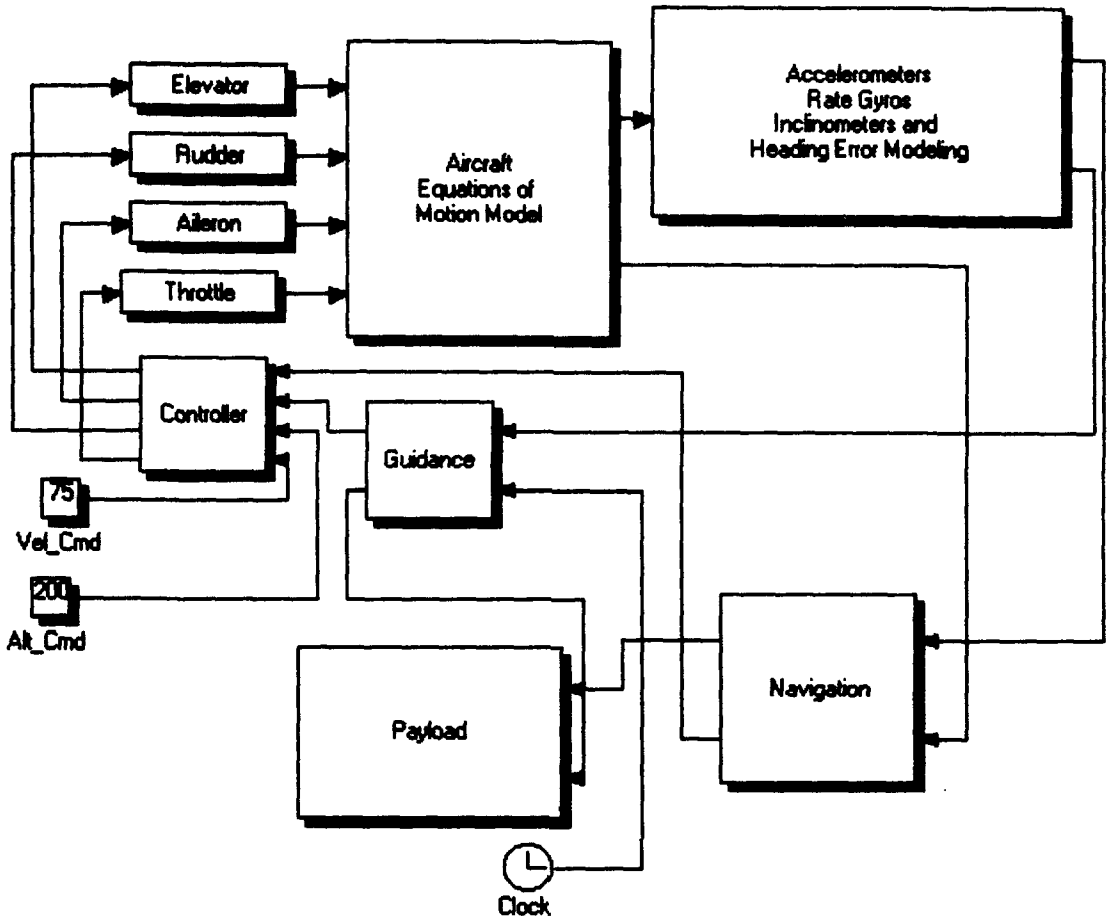


Figure 2. System Diagram

rudder surfaces and a throttle controlling a half-horsepower piston engine. Motive force is provided by a single nose-mounted tractor propeller. The Aircraft Equations of Motion Block, constructed by Kuechenmeister [Ref. 2], incorporates a nonlinear six degree of freedom equations of motion model. The stability deriva-

tives used in the model are constants either calculated or estimated for a nominal cruise flight condition of 73 feet per second airspeed at sea level altitude.

The sensor models developed in [Ref. 2] were obtained from manufacturing data used in conjunction with the outfitting of an actual NPS UAV, Archytas, with accelerometers, rate gyros, and inclinometers. Errors introduced into the sensors were obtained from the manufacturer's data for the rate gyros and accelerometers and from [Ref. 1] for the inclinometers and payload sensors . These errors were specified for the modular integrated avionics group (MIAG) architecture in the report and represent the most sensitive avionics and navigation system currently available for an unmanned aerial vehicle.

The Equations of Motion Block solves the nonlinear equations for nine states: three body-fixed translational velocities, three body-fixed rotational velocities, and three Euler angles used for rotation between a body-fixed and an inertial (earth-fixed) coordinate system. The development of these equations of motion is well covered in [Ref. 2]. The UAV's true path over the ground is determined by resolving body acceleration into inertial (earth-fixed) coordinates via a coordinate transformation and integrating the acceleration twice to obtain displacement in Cartesian coordinates. The Equations of Motion Block, since it calculates both the aircraft states and inertial position at each point in time, is the source of uncorrupted navigational and attitudinal data used as the "Truth Model" in the Monte Carlo simulation and analysis, which is discussed in Chapter IV.

The Sensor Block models accelerometers, rate gyros, and inclinometers. These "measure" body-fixed accelerations, body angular rates, and Euler Angles respectively. The models, covered in [Ref. 2], incorporate both biases and sensor noise

floor errors. Each sensor is modeled as a first-order filter with a manufacturer-supplied cutoff frequency. The accelerometer and rate gyro anti-aliasing filters were modeled as first order low-pass filters.

The Sensor Block outputs "measured" body-fixed accelerations, body angular rates, and Euler Angles. These measured parameters are passed to the Navigation Block.

B. NAVIGATION SYSTEM

The Navigation Block incorporated two different but complementary navigation systems: an inertial navigation and Global Positioning System (GPS). While GPS is setting new standards for navigational position accuracy, its slow update rate (once per second at best) makes it less than perfectly suited for rapid position updates for a fast-moving aircraft. On the other hand, an inertial navigation system, which measures body-fixed accelerations, converts them via coordinate transformation to inertial accelerations and integrates them to determine velocity and position, is good in the short term but tends to introduce position errors over time. These errors are a result of small biases in acceleration sensors. One answer to the problem is to re-initialize inertial position as often as possible. A dynamic vehicle like a UAV requires position updates many times a second, a capability inherent in inertial navigation systems. The solution to providing quick position updates and small steady state errors is to design a complementary filter which incorporates the best features of both GPS and INS.

The navigation system designed in [Ref. 3] incorporates a strapdown inertial navigation system consisting of accelerometers measuring acceleration along the aircraft x (positive forward), y (positive toward the right wing-tip), and z (positive down) axes and gyros measuring angular rates p (about the aircraft x axis), q (about y), and r (about z). Inclinometers in the sensor module are used to measure aircraft roll (phi), pitch (theta), and yaw (psi) angles. The Euler angles phi, theta, and psi are used to transform the accelerations sensed in the body coordinate system to an earth-fixed inertial coordinate system. The complete navigation system [Ref. 3] incorporated a Kalman filter to fuse the low-frequency GPS updates with the higher frequency accelerometer/rate gyro acceleration data. To improve accuracy, the system included a differential GPS model. This incorporates a ground-based system placed on a surveyed site which broadcasts its measured GPS error to the local area [Ref. 3] and is used to improve the accuracy of the onboard GPS system.

C. GUIDANCE

The guidance block in Figure 2 was developed as a part of this work and took the filtered X and Y inertial position data from the navigation block and compared it with the desired inertial position. More details are included in the next chapter.

D. CONTROLLER

The controller, developed by the author, was implemented to control four variables in steady state-airspeed, attitude, ground track heading, and rudder position. It is discussed in full in the following chapter.

E. PAYLOAD MODULE

The payload module in Figure 2 simulated a low-light television camera or FLIR used to target an enemy position. Because there was no feedback from the payload module to the rest of the model and consequently no propagation of payload sensor errors, the payload was modeled as a sensor with simple lookdown and heading errors as given for the MIAG payload in the UAV Error Analysis Report [Ref. 1]. There is currently no servo model included in the payload module.

III. CONTROLLER DESIGN, GUIDANCE, AND PAYLOAD MODEL

The controller was designed using Linear Quadratic Regulator techniques. The goal was to construct a closed-loop model which yielded satisfactory performance while quickly damping out transient responses so that steady-state errors could be easily evaluated by Monte Carlo analysis of a simulated flight. The notation adopted here means to be suggestive. Upper case letters are used to denote the full-scale values of aircraft variables or matrices, while lower case letters indicate perturbations around the trim values of the full-scale variables. A subscripted "B" indicates a variable expressed in body coordinates while a subscripted "U" indicates one expressed in inertial coordinates. A coordinate transformation from inertial to body is indicated by

$$\overset{B}{}_{U}T$$

and a body to inertial transformation is shown by exchanging the superscript and subscript.

A. PLANT DESCRIPTION

Since the controller design required a linear plant, the equations of motion model [Ref. 2] was linearized at a cruise condition of 73.3 feet per second true airspeed at a sea level standard day. The linearized model has the following form:

$$\dot{x} = Ax + Bu \qquad \text{(EQ 1)}$$
$$y = Cx + Du$$

where A and B are given in Appendix B, C is a 9x9 identity matrix, and D is a 4x9 zero matrix. The state vector x consisted of the following states:

$$x = \left[a_e\ a_r\ a_a\ a_t\ u\ v\ w\ p\ q\ r\ \phi\ \theta\ \psi\right]', \qquad \text{(EQ 2)}$$

where the first four states were the elevator, rudder, aileron, and throttle actuator positions. The states u, v, and w were linear velocities about the aircraft x (positive forward), y (positive out the right wing) and z (positive down) axes respectively. The three states p, q, and r were rotational velocities about the aircraft x, y, and z axes which used the right-hand rule to determine their orientation. The final three states, ϕ, θ, and ψ, were Euler angles used to determine the aircraft's orientation with respect to an earth-fixed coordinate system [Ref. 2]. The control input vector u is given by:

$$u = \left[\delta_e\ \delta_r\ \delta_a\ \delta_t\right]', \qquad \text{(EQ 3)}$$

where the first three elements in u represented deflections in elevator, rudder, and aileron control surfaces. The fourth element, δ_t, was measured throttle input as a percentage of total thrust available. The output vector y was the set of the measured outputs obtained from the sensor suite. The onboard sensors could not

directly measure all of the states as formed in the original model; therefore, the original set of nine states was replaced by nine measurable outputs y:

$$y = \left[a_e\ a_r\ a_a\ a_t\ v_t\ n_y\ n_z\ p\ q\ r\ \phi\ \theta\ \psi_{gt} \right]'. \quad \text{(EQ 4)}$$

The vector y was obtained by premultiplying the original set of states x by a matrix T:

$$y = Tx \quad \text{(EQ 5)}$$

Since the transformation matrix T is composed of constant elements taken about the cruise trim condition, we can also write

$\dot{y} = T\dot{x}$.

Substituting into Eq. 1, we obtain

$$T^{-1}y = AT^{-1}y + Bu. \quad \text{(EQ 6)}$$

In state space form, Eq. 6 becomes

$$\dot{y} = TAT^{-1}y + TBu. \quad \text{(EQ 7)}$$

The matrix T is called a similarity transformation. Eigenvalues for the transformed plant matrix $T^{-1}AT$ (which are the same as the eigenvalues of the original plant A) are presented in Table 1. The linearized model was stable in all modes with the exception of a slightly unstable spiral mode.

The outputs differed from the original states in the following manner:

1. Forward velocity u was replaced by true airspeed v_t.
2. Lateral and vertical velocities v and w were replaced by lateral and vertical accelerations n_y and n_z.
3. Heading ψ was replaced by ground track heading ψ_{gt}.

TABLE 1: BLUEBIRD OPEN LOOP EIGENVALUES

Mode	Eigenvalue	Damping
Short Period	-3.9173+/-3.4918i	74.6%
Phugoid	-.0057+/-.5016i	1.14%
Spiral	.0792	0
Dutch Roll	-.5322+/-3.5719i	14.7%
Roll	-5.5165	100%

By including the actuator states in the linear model, it became possible to use measurable outputs which were independent of actuator inputs. The transformation matrix is defined later in this chapter. Since there were the same number of measured outputs and original states, it was possible to design a state-feedback controller for this set of outputs.

The first transformation was effected because the pitot-static system aboard Bluebird did not measure velocity along the aircraft x-axis but rather measured the velocity in wind coordinates. This is resolved in the cruise condition that the controller was designed around by:

$$\begin{bmatrix} v_t \end{bmatrix} = \begin{bmatrix} \cos\Theta_0 & \sin\Theta_0 \end{bmatrix} \begin{bmatrix} u \\ w \end{bmatrix}, \qquad \text{(EQ 8)}$$

where Θ_0 is the trim aircraft pitch attitude of .0912 radians (approximately five degrees). Sideslip (v) and body vertical velocity (w) can be obtained practically only by integrating body y and z accelerations n_y and n_z, which are available directly from the accelerometers of the strapdown inertial navigation system. Therefore, v and w were replaced with n_y and n_z. This was accomplished by using the second and third rows of the original plant (A) matrix, since in the linear form, state derivatives can be formed by multiplying the second and third rows of the original plant matrix by the state vector,

$$\begin{bmatrix} n_y \\ n_z \end{bmatrix} = \begin{bmatrix} A(2) \\ A(3) \end{bmatrix} x, \qquad \text{(EQ 9)}$$

where A(2) and A(3) represent the second and third rows of the original plant matrix. The fourth and final transformation was from heading (ψ) to ground track heading (ψ_{gt}). The change to ground track heading, one of four variables controlled in steady state, was necessary to keep the UAV ground track always pointed in the direction of the desired waypoint. Without this transformation, the guidance routine would have continually commanded aircraft heading toward the targeted waypoint. Any crosswind would cause a cross-track error, with the aircraft following a banana-shaped route to the targeted waypoint. With the transformation to ground track heading and the controlling of rudder position to zero in steady state,

the aircraft naturally turns into the wind and flies a direct path to the next waypoint. Fortunately, ground track heading is a combination of states [Ref. 4]:

$$\psi_{gt} = \psi + \frac{\beta - \phi \sin\alpha}{\cos\gamma}. \qquad \text{(EQ 10)}$$

Equation 10 is derived in Appendix A. The newly introduced variables β, γ, and α can be computed using their steady state values at the design cruise condition. Assuming a small sideslip angle in cruise, β (sideslip) can be approximated by

$$\beta = \frac{v}{V_{T_0}} = \frac{1}{73.3} v \qquad \text{(EQ 11)}$$

Since the vehicle is trimmed in a straight and level cruise condition, γ (the difference between angle of attack, α, and inertial flight path angle, θ) is negligible. For the design cruise condition, with the aircraft in 1g, non-climbing flight, the lift L generated by the aircraft must be equal to the weight of the vehicle. Using vehicle parameters [Ref. 2] and a relation for steady-state lift from Lan and Roskam [Ref. 5],

$$L = 0.5 \rho v V_T^2 s C_{L_\alpha} \alpha = \text{Weight}. \qquad \text{(EQ 12)}$$

where ρ is air density, s is wing area, $C_{L\alpha}$ is the aircraft lift-curve slope, and α is angle of attack. Using the Bluebird parameters from Table 1 for the design cruise condition at standard temperature and pressure,

$$\alpha = 0.0912. \qquad \text{(EQ 13)}$$

Thus, Eq. 7 simplifies to

$$\psi_{GT} = \psi + \frac{1}{73.3} v - \phi \sin(0.0912). \qquad \text{(EQ 14)}$$

Now, the measured outputs can be derived from the original state vector using the 9x9 transformation matrix T:

$$\begin{bmatrix} u \\ n_y \\ n_z \\ p \\ q \\ r \\ \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \cos(\Theta_0) & 0 & \sin(\Theta_0) & 0\ 0\ 0 & 0 & 0\ 0 \\ A(2) & & & & & \\ A(3) & & & & & \\ 0 & 0 & 0 & 1\ 0\ 0 & 0 & 0\ 0 \\ 0 & 0 & 0 & 0\ 1\ 0 & 0 & 0\ 0 \\ 0 & 0 & 0 & 0\ 0\ 1 & 0 & 0\ 0 \\ 0 & 0 & 0 & 0\ 0\ 0 & 1 & 0\ 0 \\ 0 & 0 & 0 & 0\ 0\ 0 & 0 & 1\ 0 \\ 0 & \frac{1}{73.3} & 0 & 0\ 0\ 0 & -\sin(0.09) & 0\ 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \\ p \\ q \\ r \\ \phi \\ \theta \\ \psi \end{bmatrix} \qquad \text{(EQ 15)}$$

B. DESIGN REQUIREMENTS

The controller was required to make all closed loop eigenvalues of the Bluebird model stable and meet a minimum requirement of 70% damping and a maximum eigenvalue frequency of 12 radians per second. The damping requirement was mandated to prevent overshoot in system response to steady-state commands while the maximum eigenvalue requirement was present in consideration of actuator bandwidth limits. Because actuators were limited in frequency response to

12 rad/s, any system eigenvalues faster than this would have resulted in modes too fast for the control surfaces and/or throttle to deal with [Ref. 6].

C. CONTROL STRATEGY

Since only four controls (elevator, aileron, rudder, and throttle) were available, only four variables could be controlled in steady state. These controlled states were airspeed, pitch angle, ground track heading, and rudder position. Two of these states were in the aircraft x-y, or lateral, plane (ground track heading and rudder position), and two are in the x-z, or longitudinal, plane (airspeed and theta). This selection of two states in each plane was made because two of the controls, aileron and rudder, are largely lateral surfaces, and the other two, elevator and throttle, are largely longitudinal. In the design cruise condition, there is nearly zero coupling between the two planes. This will be demonstrated by the controller feedback gain matrix.

In order to keep the nose of the aircraft turned into the wind when flying its command ground track heading, rudder position was selected as one of the controlled states. Because its augmented plant is laterally stable, Bluebird is guaranteed to have a finite steady-state sideslip angle. The fact that Bluebird (and for that matter the vast majority of aircraft) is symmetrical in the body x-z plane guarantees that this sideslip angle will be zero. In order to maintain this symmetrical condition, the rudder must be commanded to zero in steady state [Ref. 4]. The other con-

trolled lateral state, ground track heading (Eq. 10), ensures that the aircraft flies a heading in the inertial X-Y plane toward the next selected waypoint.

D. SYNTHESIS MODEL

The controller synthesis model, shown in Figure 3, was formed by appending the derivative and error outputs for the variables controlled in steady state to the transformed Bluebird linear model. It served as an interface between the designer and control algorithm formed using the linearized Bluebird plant with the transformed state vector previously described. To the synthesis model were added the command inputs (rudder position, airspeed, ground track heading, and attitude). Command errors were formed by subtracting aircraft states from commanded states and integrators were placed on these errors to drive them to zero in steady state [Ref. 4]. The creation of the linear synthesis model makes possible the iterative approach used in forming the aircraft controller.

E. LINEAR QUADRATIC REGULATOR DESCRIPTION

Consider the following linear system:

$$\begin{aligned} \dot{\eta} &= A\eta + B_1 r + B_2 u \\ z &= C_1 \eta + D_1 u \end{aligned}, \qquad \text{(EQ 16)}$$

where η is composed of the measured outputs, rudder position, airspeed, attitude, and ground track heading command error states:

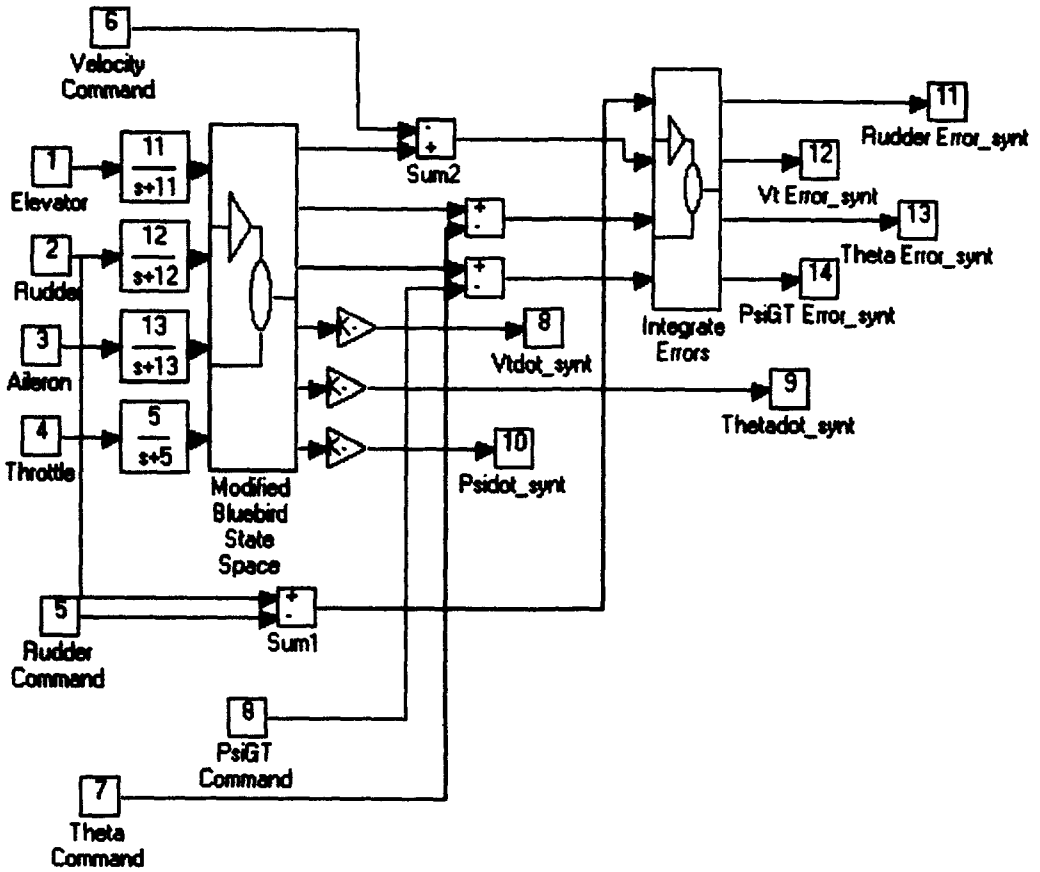


Figure 3. Bluebird Controller Synthesis Model

$$\eta = \left[y \; \delta e_r \; \delta e_a \; \delta e_\psi \; \delta e_\theta \right]'. \qquad \text{(EQ 17)}$$

Equation 16 is a state space representation of Figure 3. We assume that (A,B_2) is stabilizable, (C_1,A) is detectable, and D is full column rank. The matrix B_1 is the command input matrix, B_2 is the control surface input matrix, and r is the set of reference commands that are desired to be tracked. Now, we define a cost function,:

$$J = \frac{1}{2}\int_0^\infty \{C'QC + D'RD\}\, dt. \qquad \text{(EQ 18)}$$

where diagonal matrices Q and R, used to weight the outputs and inputs, are positive semidefinite and positive definite respectively. The Linear Quadratic Regulator (LQR) problem is to find the state feedback controller K, where

$$K = \left[K_P \; K_I \right], \qquad \text{(EQ 19)}$$

such that the feedback system shown in Figure 4 is stable and J is minimized. It can be shown that

$$K = R^{-1}B_2'P, \qquad \text{(EQ 20)}$$

where P is a stabilizing solution of the Algebraic Ricatti Equation [Ref. 10]:

$$AP + PA' - PB_2R^{-1}B_2'P + C'QC = 0 \qquad \text{(EQ 21)}$$

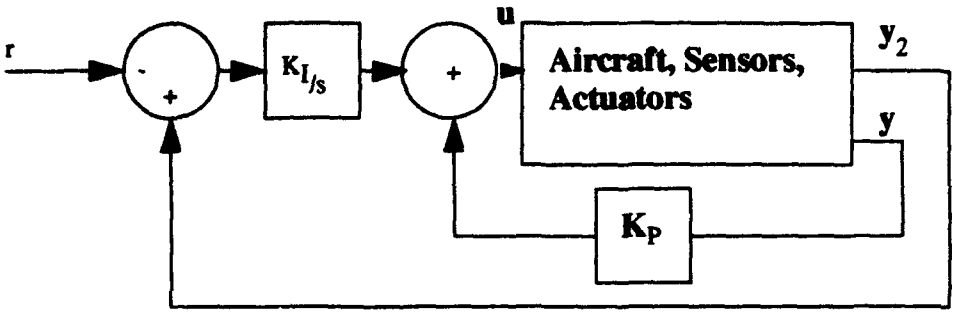


Figure 4. Feedback System

F. CONTROLLER DESIGN

As seen in Figure 3, the controller synthesis model placed an integrator on errors between measured and desired airspeed, theta, ground track heading, and rudder position. If a steady-state error exists in any of these states, then the integral of that error creates an actuating signal to drive that error to zero. Since four new error states are created by the addition of the integrators, the construction of the LQR controller creates the proper feedback gains for the error states as well as those of the transformed Bluebird model with actuators [Ref. 4]. To create a smoother transient response, the derivatives of velocity, theta, and ground track heading were also made outputs, adding them to the cost function to be described shortly. The derivative terms add a useful tool to the controller design process.

Adding weight to the state derivative outputs (via the Q matrix) increases system damping, reducing or eliminating, when damping exceeds .707, overshoot in system response [Ref. 7]. In the design of the controller, it is important to ensure that all eigenvalues of the closed loop plant, $A-B_2K$, are slower (of lower frequency) than the bandwidth of the control loops (the frequency responses between the elevator, rudder, aileron, and throttle inputs and the outputs of the feedback gain matrix of the closed loop system) [Ref. 6]. Also, since anticipated system commands in velocity, theta, and ground track heading are under 1 rad/s, command loop bandwidths (the frequency responses between command inputs and the like state outputs of the closed loop plant) in the area of 1 radian per second were considered acceptable.

The Matlab file used to create the LQR Controller, EVAL.M, is given in Appendix A. To achieve the specified design requirements, two tools available to the control designer are the Q and R matrices. By using the diagonal elements in these matrices to weight the synthesis model inputs and outputs, the designer can tune the weights iteratively to obtain desired control and command loop bandwidths, eigenvalues, and system damping. To begin design of the controller, Q and R are set as identity matrices of size 7x7 and 4x4, corresponding to the number of system outputs and control inputs respectively. Next, the maximum (highest frequency) eigenvalue of the closed loop linearized system and the minimum damping associated with the closed loop eigenvalues are recorded. After this, the frequency response of the command and control loops are observed and the associated bandwidths are observed. Finally, the time constants (the time in seconds for a state out-

put to reach 63% of a commanded step input value) for velocity, theta, and ground track headings are recorded as a measure of system performance.

Analysis of the controller begins with the control loop bandwidths: the closed-loop frequency responses between the control (elevator, rudder, aileron, and throttle) inputs and the feedback gain matrix output. These bandwidths are obtained by "cutting" the closed-loop system at the input and treating the output of the feedback gain matrix as the system output (Figure 4). This makes our system, for analysis purposes:

$$\dot{x} = (A - B_2K)\,x + B_2u$$
$$u = -Kx \qquad \text{(EQ 22)}$$

If any closed loop eigenvalue exceeds the maximum actuator bandwidth (in the case of Bluebird, 12 radians per second), then the control loop with the maximum bandwidth, which is generally responsible for the maximum eigenvalue, is penalized by increasing its respective diagonal element in the R matrix. For example, if rudder, the second input, had the widest bandwidth, the the (2,2) element in the R matrix is increased. Unfortunately, there is no formula for determining how much to penalize a control input. It is an iterative process which is repeated until no eigenvalues exceed the maximum allowable bandwidth.

The command loop bandwidths are made larger by increasing the respective diagonal elements in the Q matrix. Again, this is an iterative process in which adjustments are made, the new command bandwidths are observed, and then new adjustments are made in Q. The command loop bandwidths are observed by look-

ing at the response from the command inputs (rudder, velocity, pitch angle, and ground track heading) to the outputs of the commanded states (Figure 4)

$$\dot{x} = (A - B_2K)\,x + B_1r$$
$$y_2 = C_1x \qquad \text{(EQ 23)}$$

where C_1 is the output matrix for the four commanded states only.

The final step is the analysis of the response of the linear model to step inputs. Any overshoot can be reduced by increasing weight on the respective state derivative output, which controls damping [Ref. 6]. Obviously, the addition of too large a weight on the derivatives will slow the system response to command and control inputs.

G. BLUEBIRD CONTROLLER LINEAR RESULTS

The design iterations for the LQR controller are given in Table 2. The table presents the diagonal elements of the Q and R matrices, maximum closed loop eigenvalues, and control and command loop bandwidths.

TABLE 2: BLUEBIRD CONTROLLER SYNTHESIS

Q	R	Max E-val	Control Loop Bandwidths				Command Loop Bandwidths		
			Elev	Rud	Ail	Thr	Vel	Ψ_{GT}	Θ
[1 1 1 1]	[1 1 1 1]	-86	30	2	10	10	1	.2	.8
[1 1 1 1]	[100 1 1 1]	-44	.10	1.2	10	10	1	.2	.8
[1 1 1 1]	[100 1 1 100]	-11	10	1.2	10	1	.8	.2	.8
[1 1 100 1]	[100 1 1 100]	-11	10	1.2	10	1.5	.8	.8	.8

The feedback gain matrix is as in Eq. 19, where for the last iteration,

$$K_P = \begin{bmatrix} 0.99 & 0 & 0 & -0.00 & 0.08 & 0 & 0 & 0 & -0.53 & 0 & 0 & -3.3 & 0 \\ 0 & 0.13 & 0.40 & 0 & 0 & 0.06 & 0 & -0.07 & 0 & 3.5 & -2.4 & 0 & -1.8 \\ 0 & 0.13 & 0.24 & 0 & 0 & -0.02 & 0 & 0.1 & 0 & -1.2 & 0.92 & 0 & 0.45 \\ -0.01 & 0 & 0 & 0.49 & 0.13 & 0 & 0 & 0 & -0.18 & 0 & 0 & -3.8 & 0 \end{bmatrix} \quad \text{(EQ 24)}$$

and

$$K_I = \begin{bmatrix} -0.0992 & 0 & 0 & -0.1273 \\ 0 & -0.9923 & 0.1236 & 0 \\ 0 & 0.1236 & 0.9923 & -0.0001 \\ 0.0127 & 0 & 0 & 0.9919 \end{bmatrix} \quad \text{(EQ 25)}$$

The columns of each element of K are the feedback gains for the control actuator positions (columns 1-4), the measured outputs (columns 5-13) and the command error integrators (columns 14-17). The rows of K correspond to the elevator, rudder, aileron, and throttle commands respectively. As expected for the design

cruise condition, the longitudinal (elevator and throttle) and lateral (rudder and aileron) planes are largely uncoupled. The resulting controller has the following state space representation:

$$\begin{aligned} \dot{x}_c &= r - y_2 \\ u &= K_I x_c + K_P y \end{aligned} \qquad \text{(EQ 26)}$$

where x_c is the set of four states formed from the command errors.

The airspeed-pitch angle longitudinal combination provides a great degree of flexibility. Since

$$\theta = \alpha + \gamma \qquad \text{(EQ 27)}$$

and in the design condition,

$$\gamma = 0, \qquad \text{(EQ 28)}$$

we arrive at:

$$\theta = \alpha, \qquad \text{(EQ 29)}$$

Although angle of attack sensors are widely used and one is available on Bluebird, they are typically noisy and not well suited for feedback [Ref. 6]. In the zero flight path angle cruise condition, θ serves as a good approximation for angle of attack. From Eq. 7, which is good for 1g climbing as well as straight and level flight, for a constant airspeed, angle of attack will be constant. Any increase in θ will be reflected in a like increase in γ. If we assume a wings-level condition, then

$$V_T \sin\gamma = \dot{z} \qquad \text{(EQ 30)}$$

where z is inertial altitude. From this relation, a simple climb rate controller could be developed. To preserve simplicity and reduce computation time, this was not implemented.

The control loop response plots are shown in Figures 5-8. All control loop bandwidths were kept below the 12 rad/s limit. The line which runs horizontally near 0 dB to the respective bandwidth is the main control loop response while the other three lines in each plot are the responses of each of the three other controls to the input control.

The command loop response plots are shown in Figures 9-11. The three command loop bandwidths were placed near the target of 1 rad/s. Linear model responses to step inputs in velocity, ground track heading, and theta commands are shown in Figures 12-14. Time constants (times to reach 63% of steady state value) for these responses are 9, 4, and 8 seconds and minimal overshoot is present for each command response.

H. IMPLEMENTATION ON THE NONLINEAR MODEL

The complexity of the complete Bluebird model with sensor and GPS/INS navigation models precluded linearization with the LQR controller in the loop to determine system eigenvalues and prove stability. While the extensive system

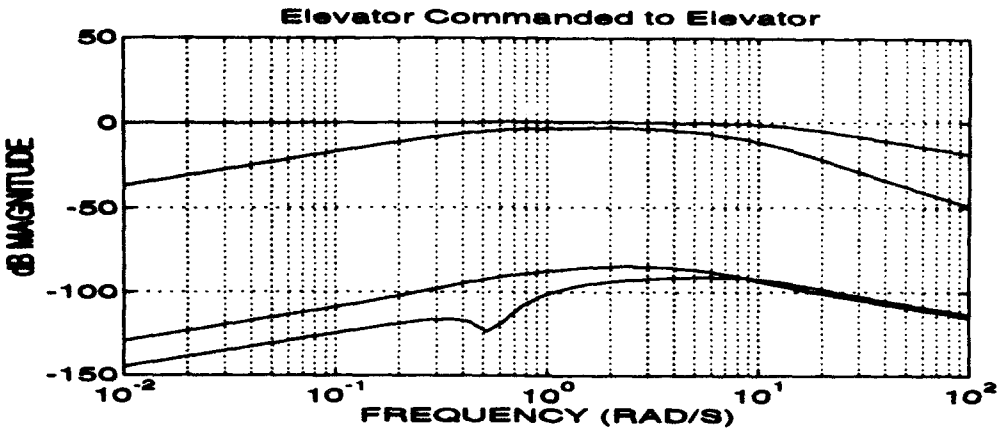


FIgure 5. Elevator Commanded to Elevator

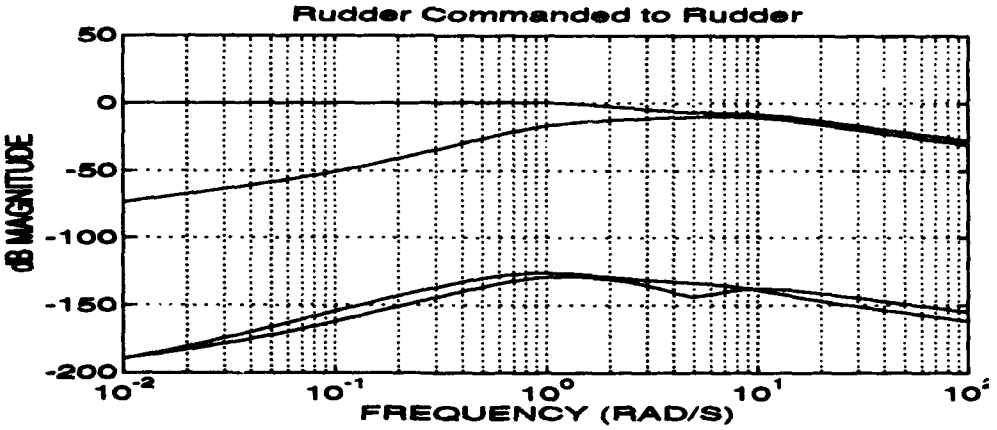


Figure 6. Rudder Commanded to Rudder

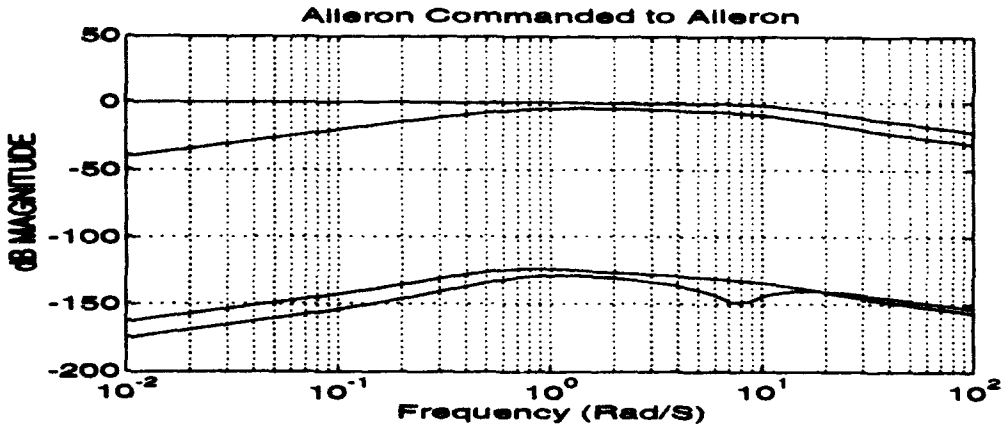


Figure 7. Aileron Commanded to Aileron

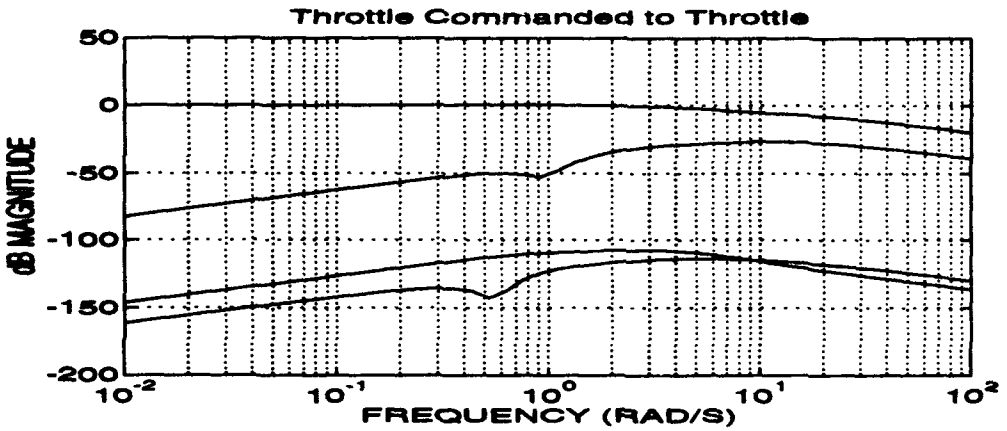


Figure 8. Throttle Commanded to Throttle

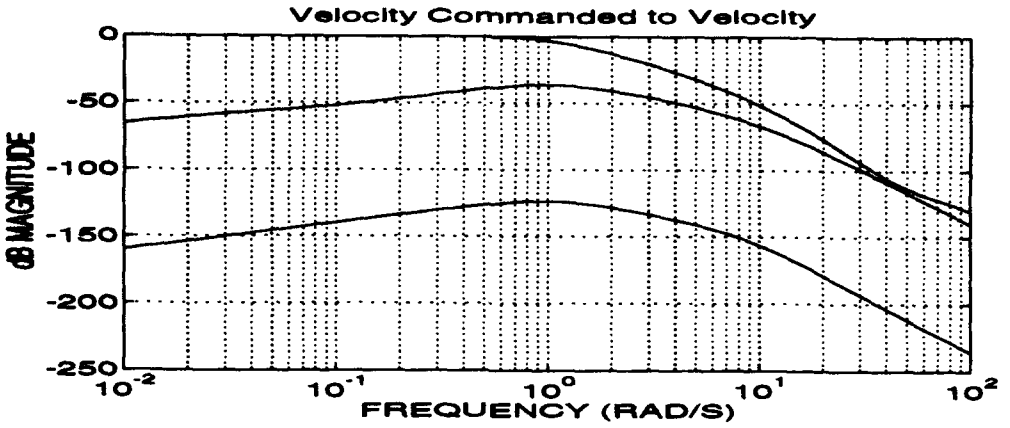


Figure 9. Velocity Commanded to Velocity

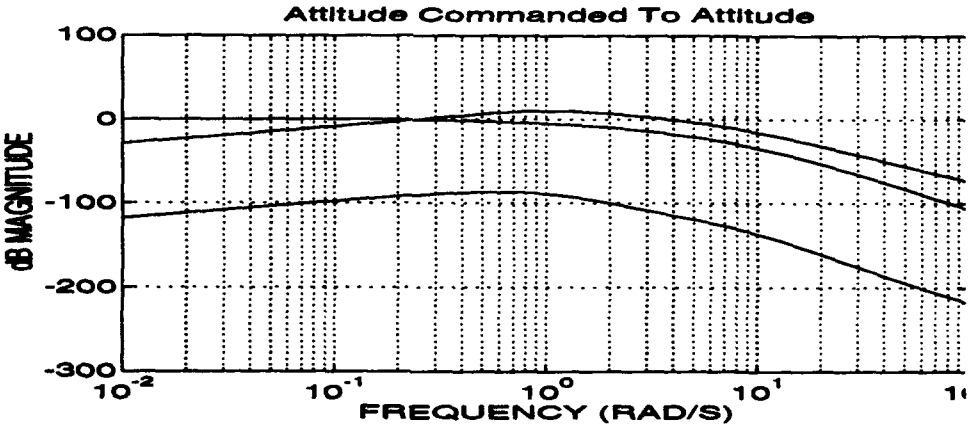


Figure 10. Attitude Commanded to Attitude

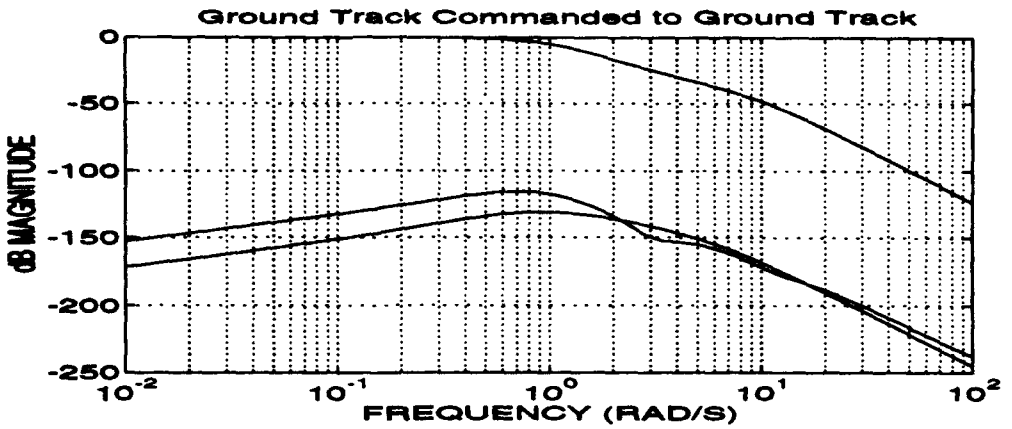


Figure 11. Ground Track Commanded to Ground Track

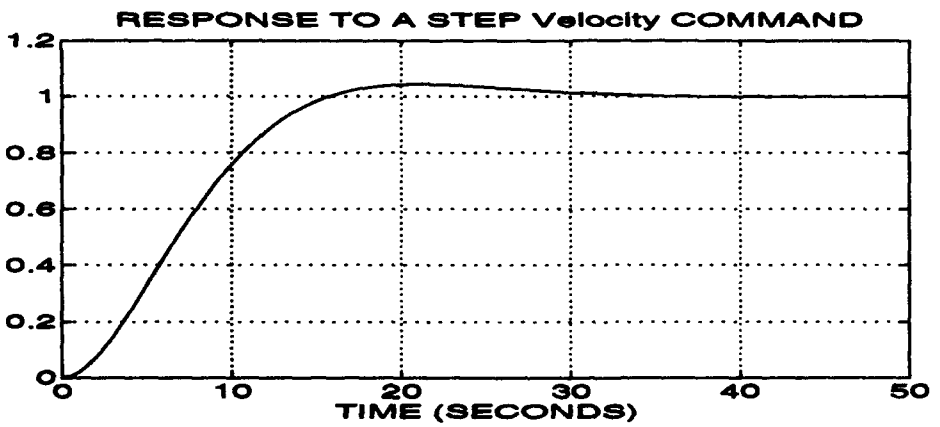


Figure 12. Response to a Step Velocity Command

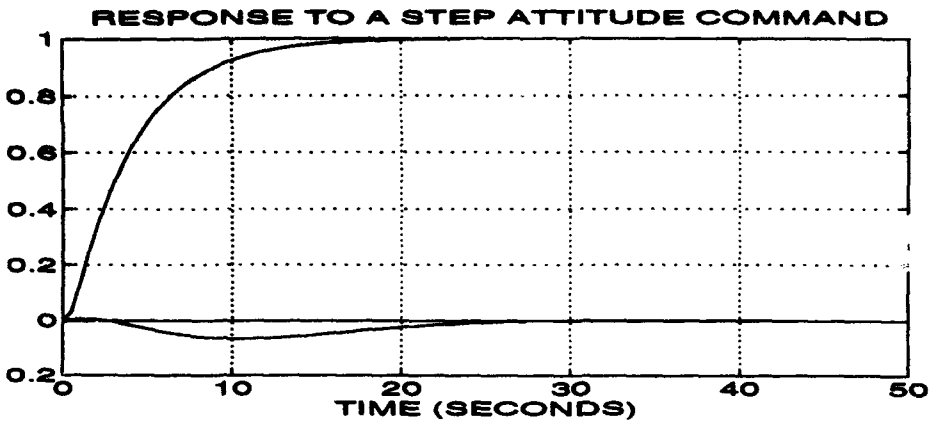


Figure 13. Response to a Step Attitude Command

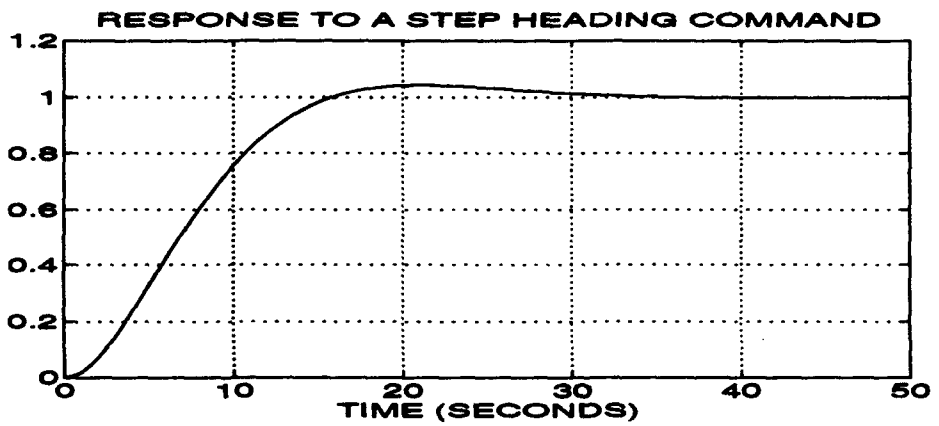


Figure 14. Response to a Step Heading Command

dynamics, including sensor models and the navigation Kalman filter, may be destabilizing, the best available indication of plant stability is that the model always returned to a stable cruise condition even when simulation was begun substantially off of the design cruise condition.

The states used in feedback were "measured" using the sensor models [Ref. 2] and the navigation filter [Ref. 3]. The airspeed sensor used the aircraft forward velocity, resolved in wind coordinates, with zero-mean white noise added to give an RMS error of .5 ft/s. The accelerometer models used to measure acceleration incorporated both gaussian zero-mean errors reflecting the modeled sensor noise floors as well as cross-axis errors equal to 0.5% of the off-axis acceleration. The rate gyro models also incorporated a noise floor with a 0.5% cross-axis error. The inclinometers incorporate a 0.1 degree RMS error with a 0.5% cross-axis error. All sensors were modeled as first-order filters with pass bands as specified by the manufacturer [Ref. 2]. The navigation filter was used to calculate inertial velocity as well as inertial position, which was fed to the guidance module for calculation of ground track heading command. Velocity and theta commands were open inputs which could be changed as the simulation was running.

The controller was implemented on the nonlinear model using Delta implementation [Ref. 8]. This involved taking the feedback integrator that was placed on the command errors (to drive them to zero in steady state) and moving it to the input of the nonlinear plant. To maintain consistency, state derivatives, rather than the states themselves, were used in feedback; the state derivatives, multiplied by the constant feedback gains were now integrated at the plant input. Recall that the controller previously obtained had the form:

$$\dot{x}_c = r - y_2$$
$$u = K_I x_c + K_P y \,. \qquad \text{(EQ 30)}$$

By moving the integrator on the command error to the plant input, the linear controller becomes:

$$\dot{x}_c = K_I (r - y_2) + K_P \dot{y}$$
$$u = x_c \,. \qquad \text{(EQ 31)}$$

Figure 15 shows the plant with the Delta-implemented controller.

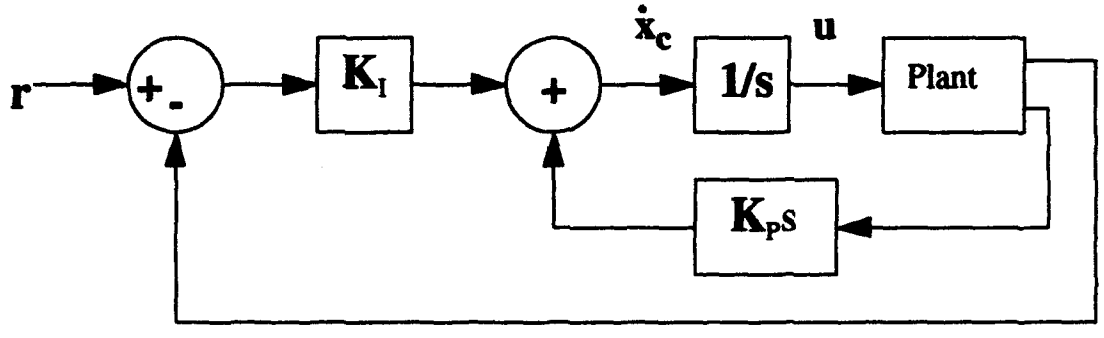


Figure 15. Controller With Delta Implementation

Further ramifications of Delta implementation are discussed in [Ref. 8]. For the purposes of this controller design, the placement of an integrator at the plant input provided an easy way to set the control surface initial conditions to their trimmed cruise values. This was important in reducing startup transients when the simula-

tion was run; limits on memory and time required that as much of each run as possible accurately reflected a model in stable flight.

I. GUIDANCE

The guidance mechanism stored three waypoints which were selected to navigate the model around a fixed target. The waypoint guidance used the error between position estimated in the navigation filter and the current waypoint to command a ground-track heading ψ_{gt} to the waypoint. This heading command was found by first taking the differences ΔX and ΔY between the estimated and the desired inertial position at each time step:

$$\begin{aligned} \Delta X &= X_{WP} - X_{EST} \\ \Delta Y &= Y_{WP} - Y_{EST} \end{aligned} \qquad \text{(EQ 32)}$$

with X_{WP} and Y_{WP} denoting the waypoint coordinates and X_{EST} and Y_{EST} being the position estimates from the navigation block. Next, the ground track heading was computed by solving the arctangent of the error components:

$$\psi_{gt} = \operatorname{atan}\frac{\Delta Y}{\Delta X}. \qquad \text{(EQ 33)}$$

When Bluebird came within two seconds of a selected waypoint, logic within the guidance block commanded the next waypoint. Because of simulation time constraints, only three waypoints were stored. Appendix A contains the Matlab code

for the guidance routine, PSICOM2.M. The Simulink guidance block is shown in Figure 16.

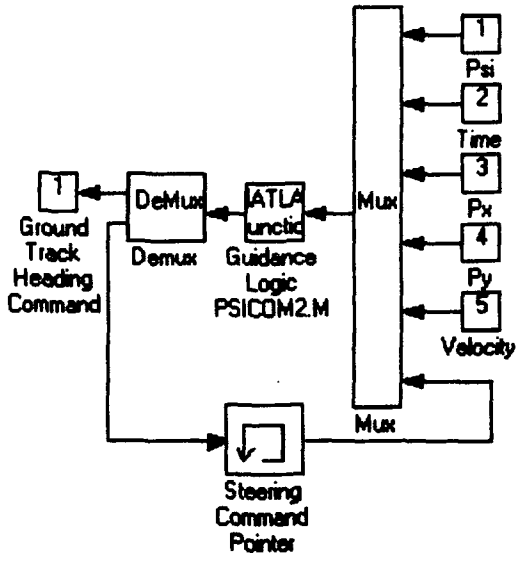


Figure 16. Guidance Block

J. PAYLOAD

The simulated payload was modeled as an open-loop sensor with azimuth and lookdown errors of 0..285 degrees at one standard deviation (1σ). Because there was no feedback from the payload sensors to the aircraft, the payload could be constructed as a simple model without loss of fidelity for the entire simulation.

The camera azimuth and lookdown angles were found in three steps. First, the differences between target position and estimated aircraft position in the inertial coordinate position (ΔX, ΔY, and ΔZ) were found. Next, the inertial differences were converted to body differences via a coordinate transformation [Ref. 2]:

$$\begin{bmatrix} \Delta x_B \\ \Delta y_B \\ \Delta z_B \end{bmatrix} = {}^{U}_{B}T \begin{bmatrix} \Delta x_U \\ \Delta y_U \\ \Delta z_U \end{bmatrix} \quad \text{(EQ 34)}$$

Finally, the body x, y, and z differences were converted to azimuth (θ_{AZ}) and look-down (θ_{LD}) angles by:

$$\theta_{AZ} = \operatorname{atan}\frac{\Delta y_B}{\Delta x_B}$$

$$\theta_{LD} = \operatorname{atan}\frac{\sqrt{\Delta x_B^2 + \Delta y_B^2}}{\Delta z_B} \quad \text{(EQ 35)}$$

The Payload Block is shown in Figure 17.

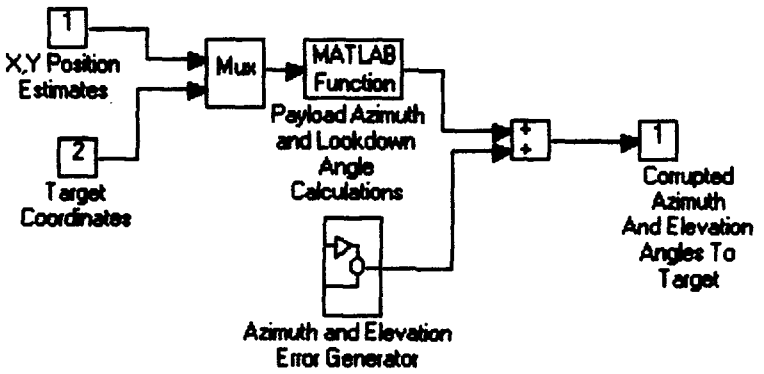


Figure 17. Payload Diagram

IV. MONTE CARLO ANALYSIS

A. MONTE CARLO OVERVIEW

The term Monte Carlo was first used to describe probabilistic mathematical methods by scientists working on the Manhattan Project in the New Mexico desert in the early 1940's. The method is essentially a series of games of chance (or random events) whose behavior and outcome can be used to study and classify mathematical properties of a random variable or a random process. By repeatedly simulating a system subject to noise, the statistical properties of the outputs of the system can be derived. Since these properties are derived from a finite number of trials, they are only an approximation of the real statistical properties of the system. As might be expected, these approximations improve with an increasing number of trials.

B. A BRIEF REVIEW OF STATISTICS

The purpose of this section is not to give an exhaustive coverage of probability and statistics, a very large topic. Rather, a brief overview of mathematical properties used in the simulation of Bluebird is presented. For a more rigorous coverage of the topic, see [Ref. 9]-[Ref. 12].

1. Probability

Given some random event A with a countable set of random outcomes $A_1, A_2, \ldots A_n$, there is associated with each outcome A_k a probability p_k between 0 and 1:

$$0 \leq p_k \leq 1 . \qquad \text{(EQ 36)}$$

If the kth outcome almost never occurs, $p_k \cong 0$; if it nearly always occurs, $p_k \cong 1$ [Ref. 9]. Note the use of the word "almost". In a stochastic process, which nearly everything is, there are no absolutes. A common notation for the probability of a random event A_k is

$$p(A_k) = p_k \qquad \text{(EQ 37)}$$

For every outcome A_k there is an associated numerical value x_k. The function assignment x_k to A_k is termed a random variable. A random process is a set of random variables indexed by time. The random, or stochastic, process analyzed in the simulation is the identification by geographical coordinates of a fixed target by the UAV. The expected value E(x) of a random variable x is defined as

$$E(x) = \sum_i p_i x_i = \mu . \qquad \text{(EQ 38)}$$

The expected value may also be thought of as the statistical mean of the random variable.

The nth central moment of x is defined as the expected value of the nth power of x:

$$E((x-\mu)^n) = \sum_i p_i (x_i - E(x))^n \qquad \text{(EQ 39)}$$

The second central moment of x is called the variance of x and is derived from Eq. 40,

$$E((x-\mu)^2) = E((x-E(x))^2). \qquad \text{(EQ 40)}$$

Substituting Eq. 40 into Eq. 42, we obtain

$$\sum_i p_i x_i^2 - E(x)^2 = E(x^2) - E(x)^2. \qquad \text{(EQ 41)}$$

The square root of the variance, σ, is called the standard deviation of x and is a measure of the dispersion of the random variable from the expected value.

A continuous, normally distributed random variable x, represented graphically by the familiar "bell curve", has the following probability density function (pdf):

$$f(x) = \frac{1}{\sigma\sqrt{2\Pi}} \exp\left|-\frac{(x-\mu)^2}{2\sigma^2}\right| \qquad \text{(EQ 42)}$$

Because errors propagate through the Bluebird simulation in a complex, nonlinear way, there are no guarantees as to the distributions of the sensor errors in the model. However, mean and standard deviation are values which are independent of distribution and so may be used to describe errors in the Bluebird simulation regardless of their distribution. Another term used in this thesis is the probability distribution function (PDF). The PDF gives the probability that a random selection of x will be less than a specified value. If the PDF is differentiable, then the pdf is defined as:

$$f(x) = \frac{d}{dx}F(x) \geq 0 \qquad \text{(EQ 43)}$$

2. Monte Carlo Fundamentals

Monte Carlo analysis is centered around the idea that integrals with infinite limits may be approximated by finite sums with a sufficiently large sample size. For a finite sum of length N and distribution G, a sampled estimate of the mean value of the random variable x, is given by

$$f(G) = \frac{1}{\sigma\sqrt{\frac{2\Pi}{N}}}\exp\left|-\frac{N(G-\mu)^2}{2\sigma^2}\right| \qquad \text{(EQ 44)}$$

As $N \rightarrow \infty$, this distribution approaches that of E(x). The observed G is within one standard error of E(x) 68.3% of the time, two standard errors 95.4% of the time, and three standard errors 99.7% of the time. This property is known as the Central Limit Theorem of probability and is substantially satisfied when

$$\left|\mu^3\right| \ll \sigma^3\sqrt{N} \quad . \qquad \text{(EQ 45)}$$

In other words, for a sufficiently large random sample N, the probability density function of the set of random samples will closely approximate the probability density function of the continuous random process. [Ref. 8]

V. SYSTEM SIMULATION

A. THE SCENARIO

The simulation involved a reconnaissance, surveillance, and target acquisition scenario. The model was first flown toward a fixed ground target; acquisition was at one nautical mile. Next, a left turn was performed in order to break closure with the target followed by a right turn to keep the UAV in close proximity to the target. For this scenario, the range to the target varied between 3000 and 6000 feet. In the data analysis, all targeting errors dependent on angular errors have been normalized by distance to the target to make them a function of angle only. The Bluebird model was "flown" over a set of three waypoints that corresponded to the desired track with respect to the target (Figure 18). Superimposed upon the trajectory in Figure 18 are the inertial X-Y position estimates from the INS/Differential GPS navigation filter. As the majority of the position error is in the inertial vertical (Z) direction, the X-Y estimates follow the aircraft track very closely. Although the simulations were started near the design cruise condition, the first three seconds of each run was not included in the evaluation in order to remove any transient system response from the analysis.

Each sensor was driven by zero mean band-limited white noise with a standard deviation obtained either from [Ref. 1] or, if it was not provided there, from [Ref. 2].

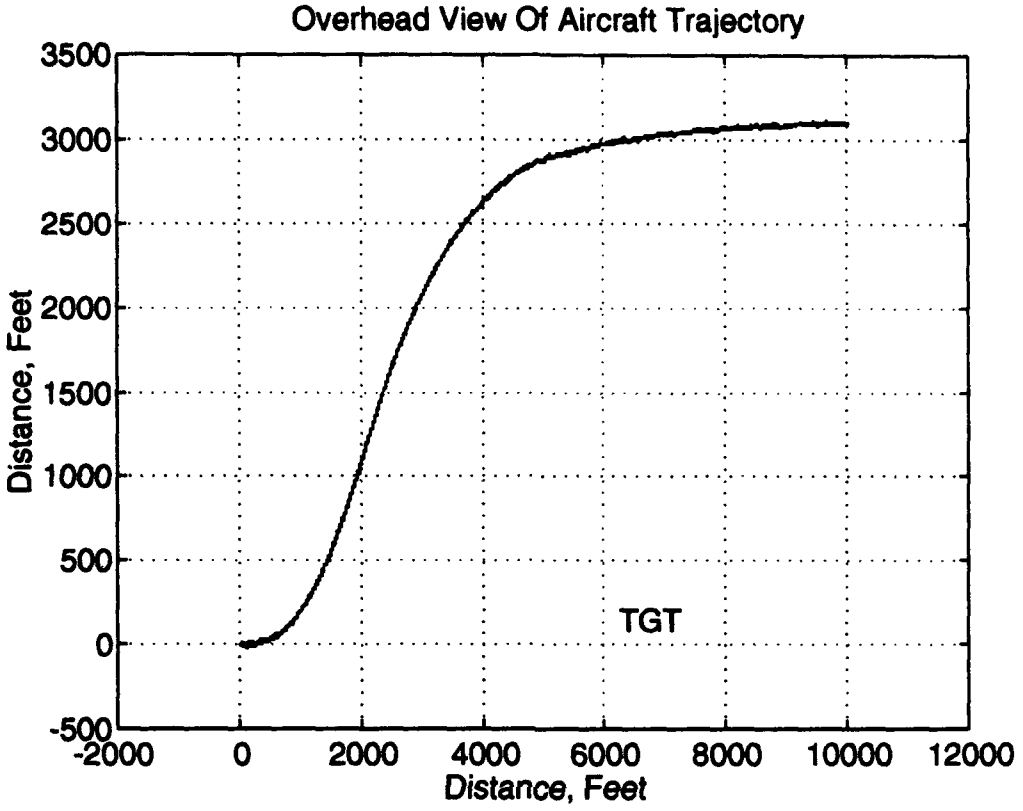


Figure 18. Model Track Over Ground

B. NUMERICAL INTEGRATION PARTICULARS

The Runge-Kutta 5 numerical integration method was used for the simulation. Although Simulink also provides a more accurate Adams-Gear integration method, use of Runge-Kutta increased the simulation speed by at least 300%. An integration step of .01 seconds was selected in order to balance system performance with fidelity. At this time interval, all noise sources in the system had to be band-limited to 50 Hertz in order to satisfy the Nyquist sampling rate:

$$2T_s \leq \frac{1}{f_{max}}, \qquad \text{(EQ 46)}$$

where T_S is the sample time in seconds and f_{max} is the maximum frequency in Hertz. Selection of a smaller time step size would have led to longer simulation time and a shorter length of data; with the .01 second step, the computer ran out of memory after approximately 150 seconds of real time, or 12.5 hours of simulation time.

C. DATA COLLECTION

Data collected for each run included the uncorrupted set of states at .01 second intervals, the measured state errors (the difference between the actual aircraft states and the measured states used in feedback), the actual inertial X, Y, and Z coordinate position, the estimated position from the navigation filter, and the payload sensor errors. The error mean values and standard deviations were calculated from the measured error data. Next, the target tracking error was calculated at each point in time for the target using the algorithm in Appendix A. The standard deviation and mean value for the error magnitude were determined across the 75 seconds of each run.

D. ANALYSIS

The sensor errors are divided between the input errors for each sensor, assumed in this case to be the manufacturer's advertised reliability of the sensor in a steady-

state, standalone sense, and the measurement errors obtained from the Monte Carlo analysis of the entire system as flown around the set of waypoints. Measurement errors were higher in the simulation data than in the static sensor values. Reasons for this included sensor dynamics, cross-track measurement errors, and the feedback nature of the system. The Simulink block diagrams used to model the inclinometers are shown in Figure 19, with the top view being the entire inclinometer block and the bottom being one of the inclinometer error blocks.

1. Sensor Dynamics

All the sensors used in the model (accelerometers, rate gyros, and inclinometers) are essentially low pass filters; their ability to measure desired quantities diminishes with the speed with which these quantities are changing. With error data from [Ref. 1] and bandwidth data from [Ref. 2], the inclinometers were modeled as first-order filters with a cutoff frequency of 2 Hertz. These inclinometers had a time constant, or time for the output to reach 63% of the input value, of 0.5 seconds; therefore, any angular change was not instantaneously measured. One way used to overcome this difficulty was to complement the low frequency inclinometers with a higher frequency device. On the Bluebird model, this was accomplished by integrating the higher frequency rate gyros and adding the integrated output to the respective inclinometer output [Ref. 3]. Both the unfiltered inclinometer output and the filtered inclinometer-rate gyro measurement were compared with the actual aircraft states to determine errors in roll, pitch, and yaw at each point in time. The resulting errors are discussed in the following chapter.

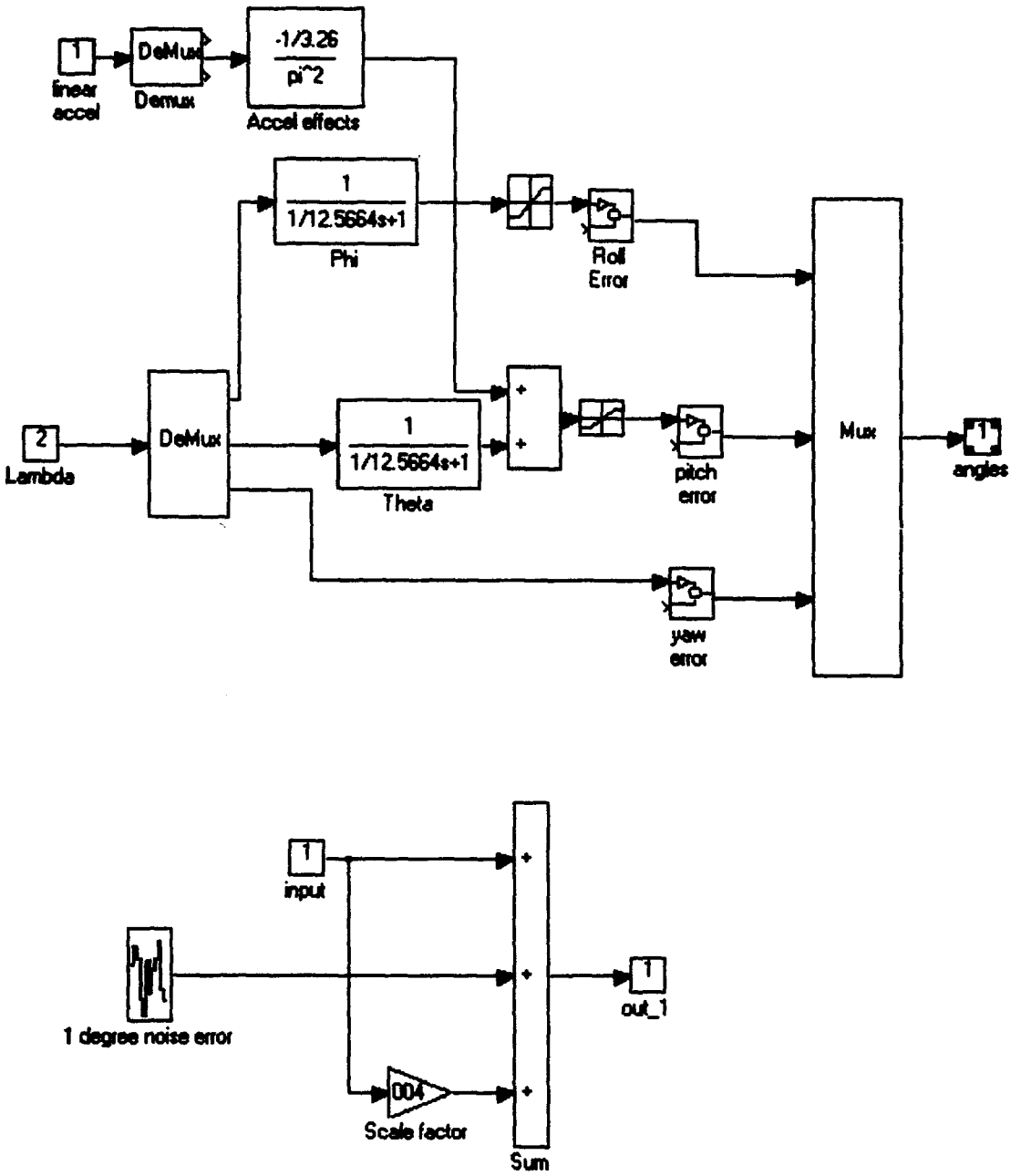


Figure 19. Inclinometer Error Models

2. Cross Axis Measurement Errors

Cross axis error refers to the false measurement signal in one channel due to an actual quantity in another channel. For example, a pure pitch rate will induce some small indication in the roll and yaw rate gyros as well. For the Bluebird model, the magnitude of cross-axis coupling was 0.5% for the accelerometers and rate gyros [Ref. 2].

3. Feedback

In an open or closed loop linear system, statistical analysis of normally distributed errors either singly or in combination is a relatively straightforward process. When one considers a closed-loop nonlinear system in which states and errors are being fed back into the system, the well-known relationships which held for the linear case are no longer valid. To analyze errors in the system, one must either form a linearized version or look at the system outputs over many trials, the so-called Monte Carlo method. The first approach, forming a linearized model for analysis, was not considered because of the inherent error introduced in linearization.

VI. RESULTS

This chapter contains results for the simulation performed in Chapter 5. Monte Carlo analysis was performed across the entire set of 7500 data points for each run to obtain the probability distribution functions for navigation and angular errors. At each time step of .01 seconds, the navigation error was obtained by taking the difference between the inertial position calculated in the Equations of Motion Block and the position estimate obtained from the Navigation Block. Sensor errors were obtained by subtracting the outputs of the sensor models from the sensor values in the equations of motion. The standard deviations of the navigation and sensor errors are shown in Table 3. The navigation errors in Table 3 are the differences between the estimated position and the position calculated in the aircraft equations of motion algorithm. The first row of the aircraft and payload sensor errors is the 1σ value of the noise fed into the system. The second row is the standard deviation calculated from the set of data points obtained using Monte Carlo Simulation.

To validate the randomness of the errors introduced into the model, the payload sensor errors, which were not subject to any of the dynamic conditions that the used vehicle sensors were, are also included in the statistical analysis of the collected data. Assuming a sufficiently large random sample, the standard deviation of the analyzed payload sensor errors should match the standard deviation of the errors introduced into the payload model, as it does.

TABLE 3: NAVIGATION AND SENSOR ERRORS

	Navigation (Feet)			Aircraft Sensors (Degrees)			Payload Sensors (Degrees)		
	X	Y	Z	Roll	Pitch	Yaw	Roll	Pitch	Yaw
Nominal Value				.10	.10	.10	.285	.285	.285
Derived Value	4.2	6.2	16.4	.118	.126	.105	.287	.284	.286

The nominal and derived sensor errors were placed in the UAV Error Analysis Program provided by NADC Warminster [Ref. 13] and a comparison was made of the error in targeting a fixed site assuming a one mile lateral standoff and 2000 feet of vertical separation from the target. The Error Analysis program assumed a flat ground plane around the target and projected a spheroid of one standard deviation of errors in inertial X, Y, and Z directions from the UAV onto the ground plane. The resulting projection formed an elliptical curve on the ground plane. Data from the error analysis program is presented in Table 4. The first column contains the noise values used in the sensor and payload models. The next column contains the values obtained from analysis of the simulation output data. In the third column are the sensitivities for each parameter (how much an error in the given parameter contributes to Circular Error Probability, or CEP). The CEP is the size of the radius placed around the target which would enclose 50% of the estimates of target position. The CEP for the nonlinear model was about 11% higher than when considering only the static sensor errors. In order to get the CEP of the static model, the UAV would have to get approximately 10% closer to the target at its present 2000 ft

altitude. From the sensor accuracies for various architectures in [Ref. 1], the model with the incorporated MIAG errors is superior in accuracy to the other architectures.

Table 4: ERROR ANALYSIS PROGRAM

ERRORS			SENSITIVITIES	CONTRIBUTION	
	NOM	ACT		NOM	ACT
Azimuth	.285	.285	32.7 m/deg	4%	3%
Elevation	.285	.285	111.6 m/deg	43%	38%
Altitude	45'	45'	.9 m/ft	6%	5%
Roll	1	1.18	79.2 m/deg	22	19%
Pitch	1	1.26	79.2 m/deg	22%	30%
Heading	1	1.05	32.7 m/deg	4%	4%
North Error	43.1 m	46.1 m			
East Error	36.8 m	42.2 m			
CEP	56.7 m	62.5 m			

Table 4 also shows how error sensitivities changed with the analysis of closed loop sensor errors. The error sensitivities are potentially very important parameters when considering various UAV architectures for target acquisition performance. From the position estimate errors and the angular measurement errors in both the aircraft and payload sensors, a Matlab routine, DATERR.M was used to calculate position error at each point in time. To do this, data for one run was first loaded into memory. At each time step, the instantaneous heading, lookdown, and roll

angles to the target Ψ_0, Θ_0, and Φ_0 were calculated from target position (x_t, y_t, z_t) and aircraft true position (x_0, y_0, z_0) using

$$\Psi_0 = \text{atan}\frac{x_t - x_0}{y_t - y_0}$$

$$\Theta_0 = \text{atan}\frac{z_t - z_0}{x_t - x_0}. \qquad \text{(EQ 48)}$$

$$\Phi_0 = \text{atan}\frac{z_t - z_0}{y_t - y_0}$$

Next, the inertial attitude errors x_a, y_a, and z_a were calculated using the relationship relating inertial distances x, y, and z to Euler angles and true distance to target R:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \cos\Psi_0 \cos\Theta_0 \\ \sin\Psi_0 \cos\Phi_0 \\ \sin\Phi_0 \cos\Theta_0 \end{bmatrix} R \; . \qquad \text{(EQ 49)}$$

For small angular errors $\Delta\psi$, $\Delta\theta$, and $\Delta\phi$, we make the approximation

$$\begin{bmatrix} x_a \\ y_a \\ z_a \end{bmatrix} = \begin{bmatrix} \cos\Psi_0 \cos\Theta_0 - \cos(\Psi_0 - \Delta\psi)\cos(\Theta_0 - \Delta\theta) \\ \sin\Psi_0 \cos\Phi_0 - \sin(\Psi_0 - \Delta\psi)\cos(\Phi_0 - \Delta\phi) \\ \sin\Phi_0 \cos\Theta_0 - \sin(\Phi_0 - \Delta\phi)\cos(\Theta_0 - \Delta\theta) \end{bmatrix} R \; . \qquad \text{(EQ 50)}$$

The navigation errors x_n, y_n, and z_n are found by simply taking the difference between the aircraft position (x_0, y_0, z_0) and its estimate:

$$\begin{bmatrix} x_n \\ y_n \\ z_n \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} - \begin{bmatrix} x \\ y \\ z \end{bmatrix}. \qquad \text{(EQ 51)}$$

The components of the navigation and attitude errors were then added to obtain the entire targeting error. Figure 20 is a plot of targeting error magnitude for one run with components normalized by distance to target. The vertical axis represents target error while the horizontal axis is time in hundredths of seconds. The Probability Density Functions for navigation error, normalized attitude error, and normalized total error are shown in Figures 21 and 22. These two graphs are for total error magnitude and do not consider error biases in any direction, which were very nearly zero for both navigation and angular errors.

An important consideration when analyzing a UAV is how onboard sensors are integrated to obtain data. In the simulation of Bluebird, angular information was obtained from both the inclinometers alone and from relatively slow inclinometers filtered with larger bandwidth rate gyro information [Ref. 3]. Table 5 presents the filtered and unfiltered standard deviations and the percent improvement for the filtered model.

Table 5: COMPARISON OF FILTERED AND UNFILTERED ANGULAR DATA

Angle Measured	Unfiltered Value, 1σ, degrees	Filtered Value, 1σ, degrees	Percent Improvement
Roll	1.24	1.04	19
Pitch	1.38	1.24	12
Heading	1.08	0.36	67

The large improvement in heading accuracy resulted because there was no delay modeled in the heading sensor [Ref. 2]. The results from Table 5 are presented in graphical form in Figure 23. For each axis, the addition of a rate sensor improved angular measurement performance. Figure 24 shows the effect on CEP of the filtered and unfiltered sensors. Not only are the accuracies of individual sensors important; the way that a sensor package is integrated has a bearing on how accurately measurements may be made. The comparison was made using [Ref. 12]. Standoff distance was one mile and vertical separation from the target was 2000 feet. Other error inputs to the program were used from the analysis values in column 2 of Table 4.

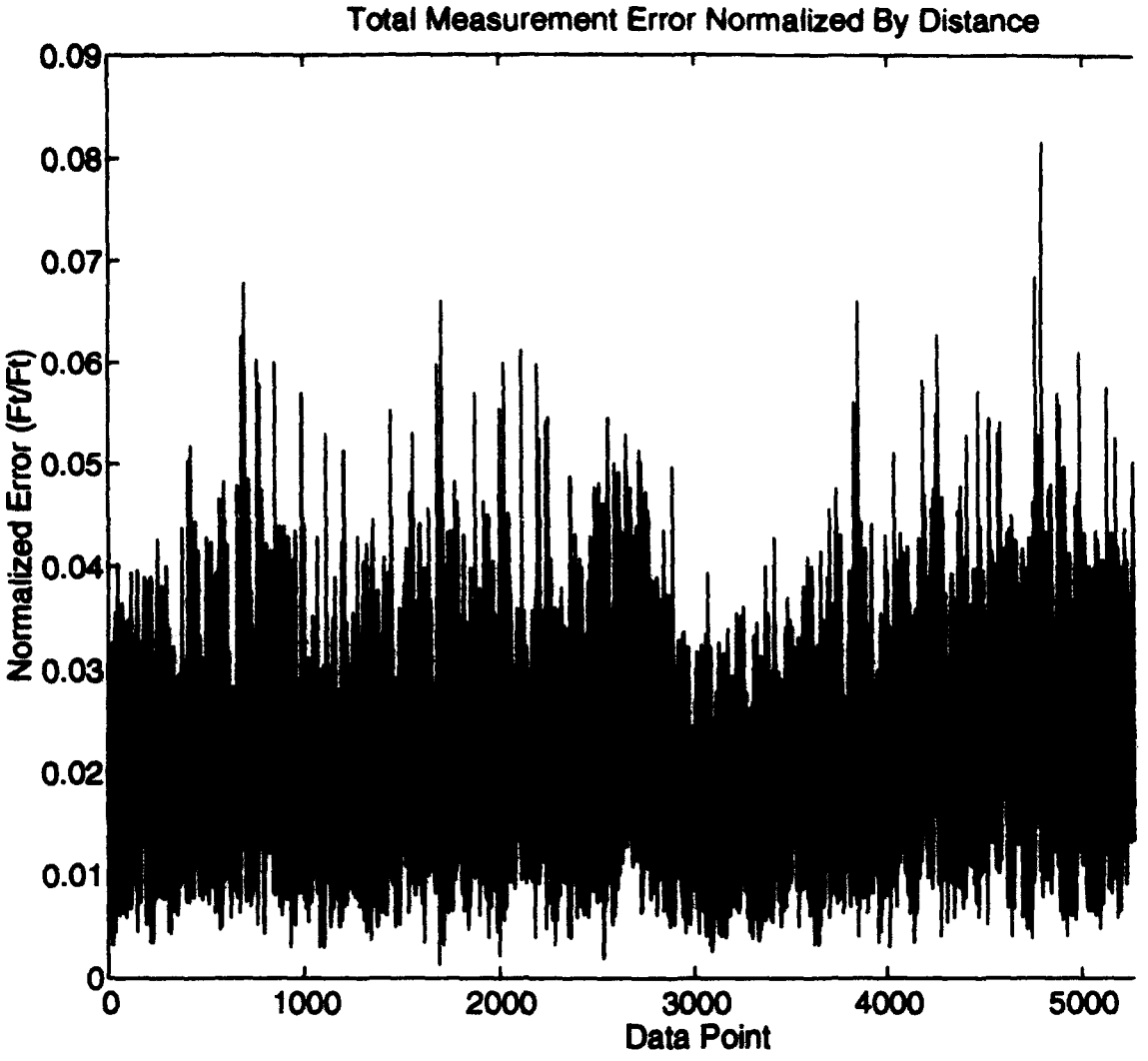


Figure 20. Targeting Error Normalized By Distance

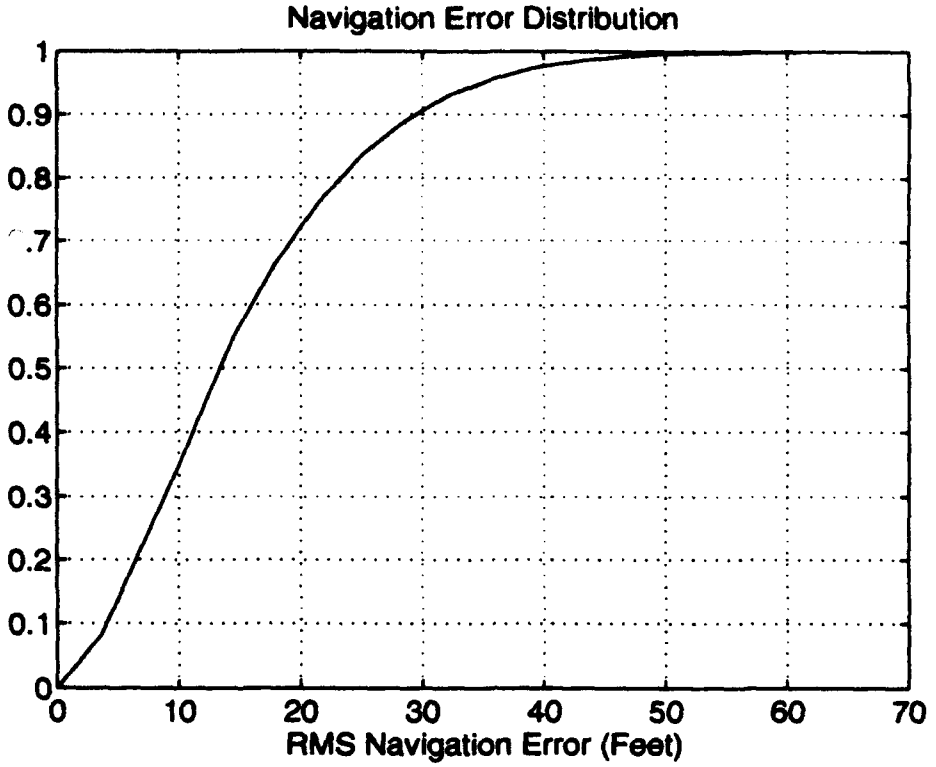


Figure 21. PDF of Navigational Errors

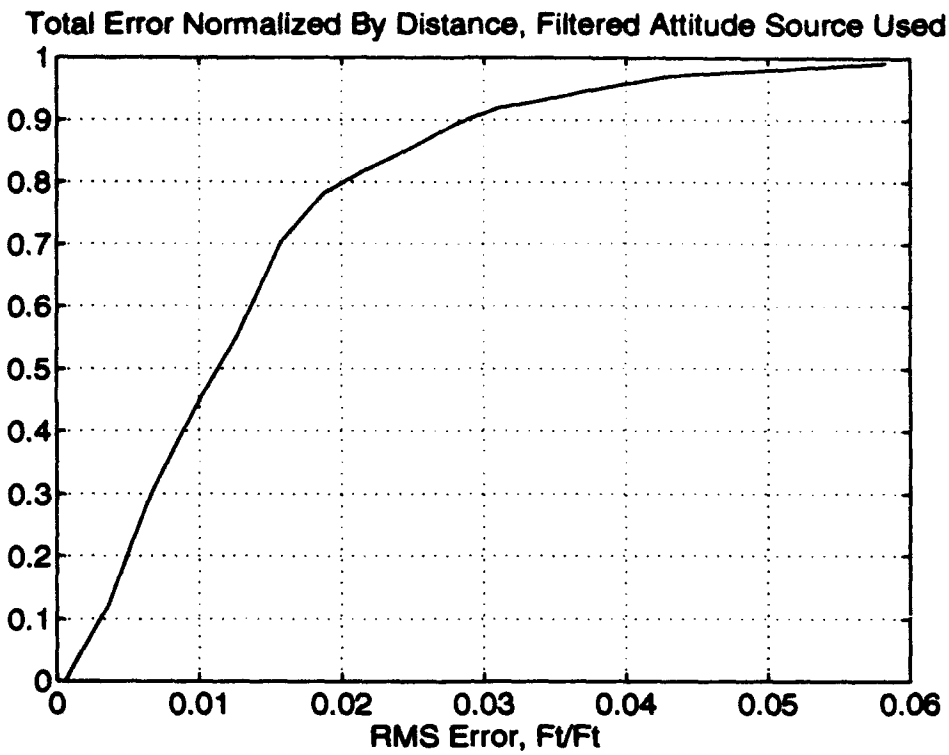


Figure 22. PDF of Attitude Errors

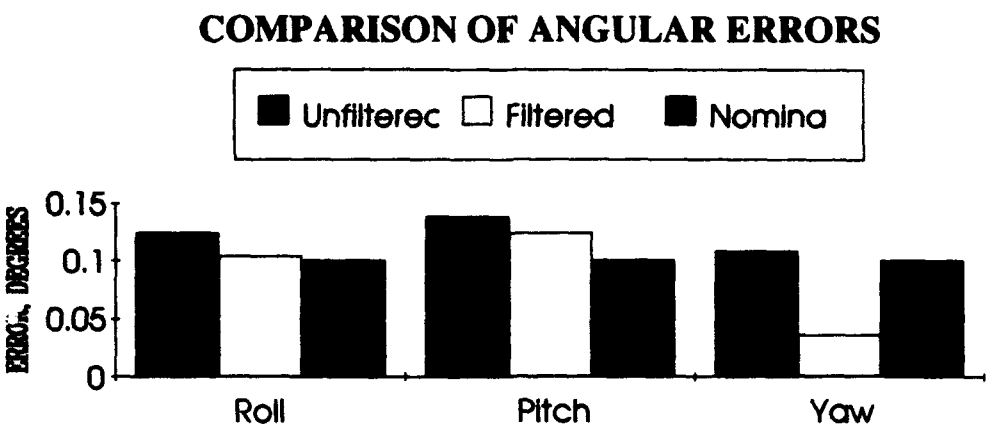


Figure 23. Angular Error Comparison

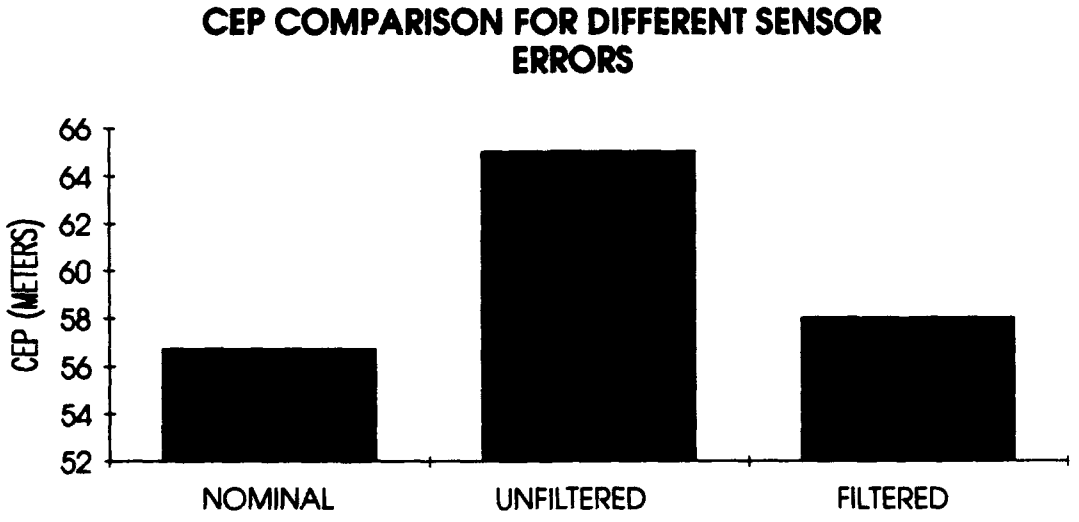


Figure 24. CEP Comparison

VII. CONCLUSIONS AND RECOMMENDATIONS

When comparing different dynamic systems with regard to how well they perform a certain mission given onboard sensors and their accuracies, it is important to look beyond how sensors perform independently in a static situation and to consider how they perform in a dynamic environment together with the rest of a vehicle's sensor package. The key to this analysis is the formulation of a high fidelity model containing all sensor and vehicle motion dynamics. Because such a model will necessarily incorporate many nonlinearities, the most general approach in accurately analyzing system performance is Monte Carlo simulation.

In the UAV model and targeting simulation considered in this thesis, it was shown that static sensor accuracies differed greatly from measured data obtained from the sensors during simulation. Also, it was important to look at what sensors were being used to provide data. For example, whether low frequency sensors were complemented with higher frequency ones to provide the vehicle with better state measurements during maneuvering flight.

The Naval Postgraduate School Avionics Laboratory has adequate facilities now to create high fidelity models and run them in realistic simulations. As the models created by successive graduate students with little programming experience become more and more complex, time constraints require that models created in a user-friendly graphical environment be able to be run in as optimal an environment as possible. The Aeronautics Department recently acquired $MATRIX_X$, a package which allows construction of models in a graphical environment similar to that of

Simulink but with an autocode feature which creates executable C code from the graphical model to allow for faster simulation.

In the near term, the model used in this paper presents a good teaching aid for both avionics system design and integration and LQR control design. For follow-on thesis students, the modularity of the UAV model allows for the modification or complete reconstruction of any or all of the subsystems. This provides an ideal environment for future studies in advanced controller design and other avionics topics. Suggested topics include:

1. The implementation of a fault detection system on the model.
2. The design of a controller based solely on inertial states.
3. The redesign of the sensor package.

The increasing cost of manned aircraft and the continuing miniaturization of avionics components will make Unmanned Aerial Vehicles more and more attractive as an alternative to airplanes and helicopters for many missions. It is incumbent that the government maintain adequate facilities to realistically evaluate future UAV designs. The Naval Postgraduate School Avionics Laboratory, with its mix of industry-experienced professors and fleet aviators, provides an excellent environment for such work.

APPENDIX A: MATLAB ROUTINES

A. GUIDANCE ROUTINE

```
% Input vector is psi, time
% outputs incremental px, py for this iteration
% also outputs place holder for guidance mechanism
%define function
function out = psicom2(u,pxc,pyc,told,ss)
%define waypoints in inertial X,Y
pxcv=[1000 6000 10000 0]
pycv=[1000 3000 3000 0]
% inputs to block are heading, time, X and Y position, velocity, and current
% waypoint
u=u'
psi=u(1)
px=u(3)
py=u(4)
speed=u(5)
tnew=u(2)
% start using waypoint at 0.2 seconds of flight time
if tnew>.2
```

```
 ss=u(6)
else ss=1
end
% define commanded waypoint
pxc=pxcv(ss)
pyc=pyc(ss)
yy=ss;
% step to next waypoint if within 2 seconds of current
perror=sqrt((pxc-px)^2+(pyc-py)^2)
if perror <= 150
 yy=yy+1
end
%  calculate ground track heading to waypoint
psic=-atan2(pyc-py,pxc-px)
out=[psic yy]'
```

B. LQR DESIGN ROUTINE

```
% load a,b matrices
[Ai,Bi,Ci,Di]=linmod('newsynt2');
% B2 is the input matrix across the actuators
B2=Bi(:,5:8);
%B1 is the command input matrix
B1=Bi(:,1:4)
```

```
% C1 is the output matrix for the three commanded states-airspeed, attitude,
% and ground track heading
C1=[0 0 0 0 1 zeros(1,12);zeros(1,11) 1 0 0 0 0 0;zeros(1,12) 1 0 0 0 0];
pause
% solve algebraic Ricatti equation to get state feedback gains
p=are(Ai,B2*inv(R)*B2',Ci'*Q*Ci);
% kstate is state feedback gains
% kerror is command error gain matrix
k=inv(R)*B2'*p;
kstate=k(:,1:13)
kerror=k(:,14:17)
% get closed loop eigenvalues
eig(Ai-B2*k);
% get damping of closed loop eigenvalues
damp(ans)
pause
% set bode frequency range from .01 to 100 rad/s
w=logspace(-2,2);
w1=w';
% solve frequency response for velocity/velocity commanded
[m1,p1]=bode(Ai-B2*k,B1,C1,zeros(3,4),2,w);
% convert to dB
m1=20*log10(m1);
mm1=[w1 m1];
```

```
save data1 mm1 /ascii
% size plot for convenient size when creating postscript file for thesis
subplot(1.3,1.3,1.2)
% full-scale bode magnitude plot for velocity/velocity commanded
semilogx(w,m1)
grid
xlabel('FREQUENCY (RAD/S)')
ylabel('dB MAGNITUDE')
title('Velocity to Velocity Commanded')
pause
% create bode magnitude plot for attitude/attitude commanded
[m2,p2]=bode(Ai-B2*k,B1,C1,zeros(3,4),3,w);
m2=20*log10(m2);
mm2=[w1 m2];
save data2 mm2 /ascii
subplot(1.3,1.3,1.2)
semilogx(w,m2)
grid
xlabel('FREQUENCY (RAD/S)')
ylabel('dB MAGNITUDE')
title('Attitude To Attitude Commanded')
pause
%
% create bode plot for ground track/ground track commanded
```

```
[m3,p3]=bode(Ai-B2*k,B1,C1,zeros(3,4),4,w);
m3=20*log10(m3);
mm3=[w1 m3];
save data3 mm3 /ascii
subplot(1.3,1.3,1.2)
semilogx(w,m3)
grid
xlabel('FREQUENCY (RAD/S)')
ylabel('dB MAGNITUDE')
title('Ground Track to Ground Track Commanded')
pause
% create bode plot for elevator/elevator commanded
[m4,p4]=bode(Ai-B2*k,B2,k,zeros(4,4),1,w);
m4=20*log10(m4);
mm4=[w1 m4];
save data4 mm4 /ascii
subplot(1.3,1.3,1.2)
semilogx(w,m4)
grid
xlabel('FREQUENCY (RAD/S)')
ylabel('dB MAGNITUDE')
title('Elevator to Elevator Commanded')
pause
% create bode plot for rudder/rudder commanded
```

```
[m5,p5]=bode(Ai-B2*k,B2,k,zeros(4,4),2,w);
m5=20*log10(m5);
mm5=[w1 m5];
save data5 mm5 /ascii
subplot(1.3,1.3,1.2)
semilogx(w,m5)
grid
xlabel('FREQUENCY (RAD/S)')
ylabel('dB MAGNITUDE')
title('Rudder to Rudder Commanded')
pause
% create bode plot for aileron/aileron commanded
[m6,p6]=bode(Ai-B2*k,B2,k,zeros(4,4),3,w);
m6=20*log10(m6);
mm6=[w1 m6];
save data6 mm6 /ascii
subplot(1.3,1.3,1.2)
semilogx(w,m6)
grid
xlabel('Frequency (Rad/S)')
ylabel('dB MAGNITUDE')
title('Aileron to Aileron Commanded')
pause
% bode plot for throttle/throttle commanded
```

```
[m7,p7]=bode(Ai-B2*k,B2,k,zeros(4,4),4,w);
m7=20*log10(m7);
mm7=[w1 m7];
save data7 mm7 /ascii
subplot(1.3,1.3,1.2)
semilogx(w,m7)
grid
xlabel('FREQUENCY (RAD/S)')
ylabel('dB MAGNITUDE')
title('Throttle to Throttle Commanded')
pause
% create time vector for time response plots
t=[0:.5:50];
t1=t';
% step response from velocity command to velocity
m8=step(Ai-B2*k,B1,C1,zeros(3,4),2,t);
mm8=[t1 m8];
save data8 mm8 /ascii
subplot(1.3,1.3,1.2)
plot(t,m8)
grid
title('Response to a Step Velocity Command')
xlabel('TIME (SECONDS)')
ylabel('RESPONSE')
```

```
pause
% step response from attitude command to attitude
m9=step(Ai-B2*k,B1,C1,zeros(3,4),3,t);
mm9=[t1 m9];
save data9 mm9 /ascii
subplot(1.3,1.3,1.2)
plot(t,m9)
grid
title('Response to a Step Attitude Command')
xlabel('TIME (SECONDS)')
ylabel('RESPONSE')
pause
% ground track heading step response to ground track input
m10=step(Ai-B2*k,B1,C1,zeros(3,4),4,t);
mm10=[t1 m10];
save data10 mm10 /ascii
subplot(1.3,1.3,1.2)
plot(t,m10)
grid
title('Response to a Step Heading Command')
xlabel('TIME (SECONDS)')
ylabel('RESPONSE')
```

C. TARGETING ERROR CALCULATION

```
% error.m
% This Matlab routine takes saved bluebird navigational data and computes
%a targeting
% error at .01 second intervals
%
% Load data from run
load d1
% Target is at (6000,0,0) feet in (X,Y,Z) universal coordinate system
xt=6000
yt=0
zt=0
% Make error computation at each step
for i=1:7500
 % Aircraft true position
   x0=P(i,10);
   y0=P(i,11);
   z0=P(i,12);
 % True heading to target
   psi0=-atan2(-y0,xt-x0);
 % True lookdown angle to target
 % Assuming 2000 ft altitude
   theta0=-atan2(z0,xt-x0);
 % True roll angle to target
```

```
phi0=atan2(z0,-y0);
% True range to target
R(i)=sqrt((3000-x0)^2+y0^2+z0^2);
%
% Get instantaneous angular errors from file
dphi=states(i,7);
dtheta=states(i,8);
dpsi=states(i,9);
%
% Calculate inertial X,Y,Z errors from attitude sensors
xerratt(i)=R(i)*(cos(psi0)*cos(theta0)-cos(psi0+dpsi)*cos(theta0+dtheta));
yerratt(i)=R(i)*(sin(psi0)*cos(phi0)-sin(psi0+dpsi)*cos(phi0+dphi));
zerratt(i)=R(i)*(sin(phi0)*cos(theta0)-sin(phi0+dphi)*cos(theta0+dtheta));
%
% Calculate inertial X,Y,Z errors from navigation
xerrnav(i)=P(i,10)-Phat(i,1);
yerrnav(i)=P(i,11)-Phat(i,2);
zerrnav(i)=P(i,12)-Phat(i,3);
%
% Sum errors in X,Y,Z
xerr(i)=xerrnav(i)+xerratt(i);
yerr(i)=yerrnav(i)+yerratt(i);
zerr(i)=zerrnav(i)+zerratt(i);
% Total error
```

```
 nerrtot(i)=sqrt(xernav(i)^2+yernav(i)^2+zernav(i)^2);
 errtot(i)=sqrt(xerr(i)^2+yerr(i)^2+zerr(i)^2);
 mnerr(i)=errtot(i)/R(i);
end
% save errors into file
save d xerr yerr zerr xerratt xernav yerratt yernav zerratt zernav errtot nerrtot
```

APPENDIX B. MATHEMATICAL PROPERTIES

A. LINEARIZED BLUEBIRD PLANT MATRICES

$$A = \begin{bmatrix} -0.0614 & 0 & 0.354 & 0 & -1.7069 & 0 & 0 & -32.0416 & 0 \\ 0 & -0.3839 & 0 & 1.8401 & 0 & -71.157 & 32.0403 & 0 & 0 \\ -0.756 & 0 & -4.681 & 0 & 66.632 & 0 & -0.0002 & -2.8771 & 0 \\ 0 & -0.1446 & 0 & -5.3294 & 0 & 1.4902 & 0 & 0 & 0 \\ 0.0159 & 0 & -0.1893 & 0 & -3.1037 & 0 & 0 & 0.0362 & 0 \\ 0 & 0.1404 & 0 & -1.0612 & 0 & -0.7884 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0.0915 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.0042 & 0 & 0 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} -4.2651 & 0 & 0 & 8.7445 \\ 0 & 5.5976 & 0 & 0 \\ -37.3493 & 0 & 0 & 0 \\ 0 & 0.621 & 45.3232 & 0 \\ -21.1695 & 0 & 0 & 0 \\ 0 & -7.0235 & -5.9720 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

B. GROUND TRACK HEADING DERIVATION

Consider an aircraft flying in straight and level flight with heading ψ. Now add a sideslip angle β. Because β is in body coordinates, it must be converted to inertial coordinates to add its effect to heading. For a cruise condition, this is accomplished by rotating β into the inertial X-Y plane by the climb angle γ:

$$\psi\cos\gamma = -\beta. \qquad \text{(EQ 47)}$$

A positive sideslip angle β corresponds to a left crab angle, while heading angle is positive to the right.

The roll angle ϕ also contributes to ground track heading. Since ϕ is measured in body coordinates as well, it must also be rotated to inertial coordinates by the climb angle. It contributes to ψ with angle of attack. A positive roll angle corresponds to a right wing down condition.

$$\psi\cos\gamma = \phi\sin\alpha \qquad \text{(EQ 48)}$$

Heading angle increases to the right when viewing the inertial X-Y plane from above. Since sideslip and roll angle are used to counteract drift, they contribute to ground track heading positively and negatively respectively. Therefore, the relationship used to obtain ground track heading becomes:

$$\psi_{gt} = \psi + \frac{\beta - \phi\sin\alpha}{\cos\gamma} \qquad \text{(EQ 49)}$$

C. SIMULATION STARTUP PROCEDURES

1. Logon
2. In UNIX Command Window, type "xhost +"
3. Set environment for the machine in use:

"setenv DISPLAY 131.120.149.xx" where xx is a machine-specific code.

4. type "matlab"

5. Change to the appropriate directory.

6. type "birdset". This is a .M file which loads startup variables into the work-space and calls the Simulink program BIRD_NL5.M.

7. From the BIRD_NL5.M pulldown menu, select "Simulation", then "Start".

8. Variables saved to the workspace include P (actual positions and states), Phat (estimated positions), and states(differences between estimated and actual states).

9. Simulation runs on Hornet (a Sparc10 workstation) at a rate of one second of real time for every five minutes of real time.

REFERENCES

1. Department Of The Navy, Naval Air Development Center Warminster, *UAV Error Analysis Report*, January 1992.
2. Kuechenmeister, David R., "A Non-Linear Simulation For An Autonomous Unmanned Aerial Vehicle," *Master's Thesis*, Department Of Aeronautics, Naval Postgraduate School, Monterey, CA, September 1993.
3. Marquis, Carl, " ", *Master's Thesis*, Department Of Aeronautics, Naval Postgraduate School, Monterey, CA, September 1993.
4. Kaminer, I.I., Khargoner, P.P., and Robel, G., "Design of Localizer Capture and Track Modes For a Lateral Autopilot Using $H\infty$ Synthesis, *IEEE Control Systems Magazine*, vol 10, pp. 13-21, 1990.
5. Roskam, Jan, and Lan, Edward, *Airplane Aerodynamics And Performance*, Roskam Aviation And Engineering Corp., 1980.
6. Kaminer, I.I., *Class Notes For AE4276*, Naval Postgraduate School, 1993.
7. Rowland, James R., *Linear Control Systems-Modeling, Analysis, And Design*, John Wiley and Sons, Inc., 1986.
8. Kaminer, I.I., Pascual, A.M., and Khargonekar, P.P., "A Velocity Algorithm For Implementation Of Non-Linear Gain-Scheduled Controllers", submitted for publication in *Automatica.*
9. Kalos, Malvin H., and Whitlock, Paula A., *Monte Carlo Methods, Volume I: Basics*, John Wiley and Sons, Inc., 1986.

10. Spiegel, Murray R., *Theory And Problems Of Probability And Statistics*, McGraw-Hill, Inc., 1975.

11. Shakarian, A., "Application of Monte Carlo Techniques To The 757/767 Auto-land Dispersion Analysis By Simulation", Boeing Commercial Airplane Company, Seattle, WA.

12. Peebles, Peyton Z., *Probability, Random Variables, And Random Signal Principles*, McGraw-Hill, Inc., 1987.

13. Naval Air Development Center Warminster, PA, *UAV Error Analysis*, MS-DOS Software, 1992.

INITIAL DISTRIBUTION LIST

	No. of Copies
1. Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2. Library, Code 52 Naval Postgraduate School Monterey, California 93943-5002	2
3. Dr. Isaac I. Kaminer Department of Aeronautics and Astronautics, Code AA/Ka Naval Postgraduate School Monterey, California 93943-5000	5
4. Lt. Joseph P. Fordham 370 Cross Hill Road Monroe, Connecticut 06468	2